Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Informatietechnologie

Ontdekken van en groepscommunicatie met toestellen met beperkte mogelijkheden in het 'internet der dingen' door middel van het 'Constrained Application Protocol'

Discovery and Group Communication for Constrained Internet of Things Devices using the Constrained Application Protocol

Isam Ishaq

Proefschrift tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen:
Computerwetenschappen
Academiejaar 2015-2016

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Informatietechnologie

Promotoren: prof. dr. ir. Jeroen Hoebeke
prof. dr. ir. Ingrid Moerman

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur

Vakgroep Informatietechnologie
Gaston Crommenlaan 8 bus 201, B-9050 Gent, België

Tel.: +32-9-331.49.00
Fax.: +32-9-331.48.99

Proefschrift tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen:
Computerwetenschappen
Academiejaar 2015-2016

Acknowledgements

Although this dissertation carries my name on its cover, it would have never been possible to complete without the support and help of many wonderful people. I apologize in advance that without doubt I will forget some people. I offer my regards and blessings to all of those who supported me in any respect during the completion of this PhD research. However, I do have to mention certain persons by name and it is my pleasure to do so.

First and foremost, I would like to express my sincere gratitude to my promotors Prof. Jeroen Hoebeke and Prof. Ingrid Moerman for the continuous support of my PhD research, for their patience, motivation, and immense knowledge. Their guidance helped during the research and the writing of this dissertation. I could not have imagined having better advisers and mentors for my PhD research.

Beside my promotors, I would like to thank the jury members of my PhD: Dr. Antonio Jara, Dr. Peter van der Stok, Prof. Kris Steenhaut, Prof. Hendrik Van Landeghem, Prof. Erik Mannens, and Prof. Eli De Poorter for going through my dissertation in great detail and providing excellent comments and suggestions for improvements.

My sincere thanks also goes to Prof. Piet Demeester, who provided me the opportunity to join his team as a researcher, and who gave access to the laboratory and research facilities. Without his precious support, it would not be possible to conduct this research.

This PhD research and our stay in Ghent would not have been possible without the logistic and financial support of VLIR, Ghent University, and Al-Quds University. It is an honor for me to thank these respectful organizations for their role in encouraging research and development. Specifically I like to thank Prof. Henri Verhaaren, Prof. Tammy Schellens, Prof. Rashid Jayousi, Prof. Badie Sartawi, Prof. Tamer Essawi, and Mr. Ibrahim Abu Kteish for leading the joint Belgium-Palestinian project ELDIR-med, which enabled me to start this PhD research. I also like to thank the current and former administration of Al-Quds University for providing me the opportunity to pursue my PhD: Prof. Sari Nusseibeh, Prof. Imad Abu Kishek, Prof. Hassan Dweik, Prof. Said Zidanni, Prof. Hanna Abdel-Nour, and Mr. Ghassan Aldeek.

I thank all my colleagues at IBCN for the stimulating discussions and for all the fun we have had in the last five years. I am indebted to many of them for the support they have provided along the way to finalizing the PhD. In particular, I am grateful to Jen Rossey, Floris Van den Abeele, and Peter Ruckebusch for the numerous occasions where they helped discuss and debug errors. I am also grateful

to Bart Jooris, Pieter Becue, and Vincent Sercu for their help with the testbeds and to Michael Mehari for the help in sniffing WSN traffic. A special thank you to Pieter Willemen, Floris, and Jen for translating the summary of this dissertation into Dutch – I know it was not easy.

I am heartily thankful to my current and previous colleagues and friends in office 3.17 for being such a nice group of persons to be around for the major part of the day: Daan, Dries, Elnaz, Enri, Floris, Girum, Irina, Jen, Maarten, Michael, Milos, Opher, Pieter, Tarik, Tom, Wei, and Wim. I am also thankful to the rest of the wireless and mobile group at IBCN: Bart, Christophe, Jan, Jetmir, Jono, Merima, Peter, Pieter, Stefan, Vasileios, Vincent, and Waqar. During the time I have spent at IBCN I have witnessed the IBCN group becoming an increasingly diverse multi-cultural team. Many thanks to Thomas Demeester for organizing several meetings in which non-Dutch speaking IBCN members had a chance to practice it and know a bit more about the Flemish people and about each other. Thank you Domenico, Krishnan, Farhan, Sachin, Sahel, and the many others who shared their experience and parts of their diverse cultures with rest of us. Research alone does not make the IBCN wheel go around. I like to express my thanks to the administrative and support staff who make sure it is going around smoothly: Bernadette, Bert, Davinia, Joeri, Jonathan, Martine, Sabrina, and Simon.

The past five years were not just about completing this PhD research alone. However, many things in our private lives had to be adapted to make it possible. The biggest private challenge was the decision to relocate with my dear wife and lovely kids to Belgium or stay in Palestine. We relocated and naturally had to deal with too many (sometimes, not so) little things. Looking back at it, I am glad we made this decision since the time we spent in Ghent was very nice and full of new (mostly positive) experiences. All of us made new friends, learned Dutch (at various levels, me being at the lowest), and enjoyed living among the Flemish people and the many internationals in this beautiful city. Many people contributed to making our stay in Ghent such a pleasant one, to whom I would like to show my gratitude. Lieve, Henri, Seppe, and Yvonne for helping us settle in without too many hassles and for the many other things. Mr. Alain de Vlaeminck and all the teachers of the IVG school of Ghent for providing such a welcoming atmosphere at the school that made our kids integrate seamlessly in the school life in a very short period. Omar and Jihad for the many tips that helped us getting many things done more efficiently. Omar, Nancy, Yousef, Rayan, Dima, Robert, Rose, and Miryam for being friends of our family and for the nice times we spent together during holidays and vacations. George, Patrick, Julian, Naji, and Fadi for the wonderful vacation in Berlin – It was nice to visit the German capital with my family and let them meet my old friends.

Last but not least, I would like to thank my family: my mother Laila and my sisters Maram, Ruba, and Diala. They have always been there for me with their constant love, care and support, without which I would not have become the person I am now. My late father has left us too early, but his memory will always be my true guide in this world. I am also grateful for my extended family: my in-laws, aunts, uncles, and cousins who have always been there when I have needed them.

This acknowledgment is incomplete without thanking the four persons that have had drastic changes to their lives and who have sacrificed most because of this PhD. My daughter Jude and two sons Siraj and George, who had to be taken away from their friends and accustomed environment and had to learn a new language and form new friendships. Above everyone else, I owe my deepest gratitude to my lovely wife Suhair for being there for all of us and for supporting me spiritually throughout writing this dissertation and for sharing my life for better or for worse.

Ghent, September 2015
Isam Ishaq

Table of Contents

Acknowledgements i

Samenvatting xxiii

Summary xxvii

1 Introduction 1
- 1.1 Context . . . 1
  - 1.1.1 The Internet of Things . . . 2
    - 1.1.1.1 IoT application domains . . . 3
    - 1.1.1.2 IoT things . . . 5
    - 1.1.1.3 IoT networks . . . 7
  - 1.1.2 Open standards stack for the IoT . . . 8
- 1.2 Challenges . . . 12
- 1.3 Outline . . . 13
- 1.4 Research contributions . . . 16
- 1.5 Publications . . . 17
  - 1.5.1 Publications in international journals (listed in the Science Citation Index) . . . 18
  - 1.5.2 Publications in other international journals . . . 18
  - 1.5.3 Publications in international conferences (listed in the Science Citation Index) . . . 18
  - 1.5.4 Publications in other international conferences . . . 19
  - 1.5.5 Contributions to standardization bodies . . . 19
  - 1.5.6 Patent applications . . . 20
- References . . . 21

2 Enabling the Web of Things: Facilitating deployment, discovery and resource access to IoT objects using embedded web services 23
- 2.1 Introduction . . . 24
- 2.2 From IoT to WoT . . . 26
- 2.3 Problem statement . . . 28
- 2.4 Solution . . . 30
  - 2.4.1 Assumptions . . . 30
  - 2.4.2 The solution . . . 31

2.4.3 HTTP access to sensors . . . 34
2.5 Next steps: basics for realizing Web of Things . . . 35
2.6 Deployment . . . 37
2.7 Analysis of the different approaches . . . 41
2.7.1 Pull approach . . . 42
2.7.2 Push approach . . . 45
2.7.3 Pull-Push approach . . . 46
2.8 Comparison with related work . . . 47
2.8.1 Related work . . . 47
2.8.2 High-level comparison . . . 48
2.9 Conclusions . . . 49
References . . . 52

3 Flexible Unicast-Based Group Communication for CoAP-Enabled Devices 57
3.1 Introduction . . . 58
3.2 CoAP overview . . . 61
3.2.1 Base CoAP . . . 61
3.2.2 Resource discovery . . . 63
3.2.3 Blockwise transfer . . . 64
3.2.4 Group communication . . . 66
3.3 Group communication requirements . . . 66
3.4 Existing solutions . . . 68
3.5 Group communication using unicasts . . . 71
3.5.1 System overview . . . 71
3.5.2 Entity creation . . . 73
3.5.3 Validation process . . . 74
3.5.4 Entity usage . . . 79
3.5.5 Entity modification and behavior manipulation . . . 81
3.6 Implementation and evaluation . . . 81
3.6.1 Implementation . . . 81
3.6.2 Functional evaluation . . . 84
3.6.3 Performance evaluation . . . 87
3.6.3.1 Experiment setup . . . 87
3.6.3.2 Congestion control optimizations . . . 90
3.6.3.3 Reliability . . . 92
3.6.3.4 Number of packets . . . 95
3.6.3.5 Response time . . . 97
3.6.3.6 Impact of caching . . . 98
3.6.3.7 Entity validation overhead . . . 101
3.7 Discussion . . . 104
3.8 Conclusions/Outlook . . . 106
References . . . 107

4 Observing CoAP Groups Efficiently **111**
4.1 Introduction . . . 112
4.2 Related work . . . 115
4.3 CoAP overview . . . 118
4.3.1 Base CoAP – RFC 7252 . . . 118
4.3.2 Observing resources . . . 120
4.3.2.1 Observe option . . . 120
4.3.2.2 Observe consistency model . . . 121
4.3.2.3 Observe extension to Web Linking . . . 122
4.3.3 Group communication . . . 122
4.4 Flexible and observable entities . . . 123
4.4.1 Group communication using unicasts . . . 123
4.4.1.1 System overview . . . 123
4.4.1.2 Entity creation . . . 126
4.4.1.3 Entity usage . . . 126
4.4.2 Group observation . . . 128
4.5 Optimizing the number of client notifications . . . 129
4.5.1 Entity operation . . . 130
4.5.2 Entity precision . . . 132
4.5.3 Notifications aggregation window . . . 135
4.6 Implementation and evaluation . . . 137
4.6.1 Implementation . . . 137
4.6.2 Functional evaluation . . . 138
4.6.3 Performance evaluation . . . 139
4.6.3.1 Entity operation . . . 141
4.6.3.2 Entity precision . . . 143
4.6.3.3 Notifications aggregation window . . . 144
4.6.3.4 Combination of properties . . . 147
4.7 Conclusions and outlook . . . 147
References . . . 150

5 Experimental Evaluation of Unicast and Multicast CoAP Group Communication **155**
5.1 Introduction . . . 156
5.2 Motivation and requirements . . . 158
5.2.1 Need for group communication in the IoT . . . 158
5.2.2 Use case: Building Automation Systems . . . 159
5.2.2.1 Types of connected devices . . . 159
5.2.2.2 Grouping . . . 160
5.2.3 Requirements . . . 161
5.3 CoAP group communication . . . 162
5.3.1 The Constrained Application Protocol (CoAP) . . . 163
5.3.2 Multicast group communication . . . 164
5.3.3 Unicast group communication . . . 165
5.3.4 Hybrid group communication . . . 166

5.3.4.1 Entity profile . . . 167
5.3.4.2 Group membership management . . . 168
5.3.4.3 Summary . . . 168
5.4 Implementation . . . 169
5.4.1 Multicast group management using CoAP . . . 169
5.4.2 EM multicast extensions . . . 170
5.5 Evaluation . . . 171
5.5.1 Functional evaluation . . . 172
5.5.1.1 Multicast entities . . . 172
5.5.1.2 Extended entity features . . . 174
5.5.2 Performance evaluation on a wireless sensor testbed . . . 175
5.5.2.1 Congestion control optimizations . . . 176
5.5.2.2 Reliability . . . 181
5.5.2.3 Response time . . . 182
5.5.2.4 Group size . . . 183
5.5.2.5 CoAP retransmission timeout . . . 185
5.5.3 Evaluation in a real life setup . . . 186
5.6 Related work . . . 189
5.7 Conclusions and outlook . . . 191
References . . . 193

6 Conclusion **197**
6.1 Summary and conclusions . . . 198
6.2 Outlook . . . 200

A IETF Standardization in the Field of the Internet of Things (IoT): A Survey **203**
A.1 Introduction . . . 204
A.2 Integration of constrained devices into the Internet . . . 205
A.3 IEEE 802.15.4 . . . 208
A.3.1 Physical Layer . . . 208
A.3.2 MAC Sublayer . . . 209
A.3.3 IEEE 802.15.4 based solutions . . . 210
A.4 IETF 6LoWPAN Working Group (IPv6) . . . 211
A.4.1 Key protocols . . . 212
A.4.1.1 6LoWPAN Frames . . . 212
A.4.1.2 Header compression . . . 214
A.4.1.3 Fragmentation . . . 217
A.4.1.4 Mesh-Under Routing support . . . 217
A.4.2 Implementation and evaluation . . . 219
A.4.2.1 Implementation . . . 219
A.4.2.2 Evaluation . . . 220
A.4.3 Leveraging upon 6LoWPAN to realize the IoT . . . 221
A.4.3.1 Improvements to core specifications . . . 221
A.4.3.2 6LoWPAN over Non IEEE 802.15.4 technologies 222

A.4.3.3 Adoption of 6LoWPAN in real life use cases . . 222
A.4.3.4 Other efforts 223
A.4.4 Research challenges 223
A.5 IETF ROLL Working Group 224
A.5.1 Group description and key protocols 224
A.5.1.1 Description 224
A.5.1.2 IPv6 Routing Protocol for Low Power and Lossy Networks 225
A.5.2 Implementation and evaluation 228
A.5.2.1 Implementation 228
A.5.2.2 Using the protocol 229
A.5.2.3 Sensor-to-Sensor traffic 229
A.5.2.4 Multipoint-to-Point traffic 230
A.5.2.5 Multicast 230
A.5.2.6 Anycast 230
A.5.2.7 Link estimation 231
A.5.2.8 General performance 231
A.5.3 Leveraging upon RPL to realize the IoT 232
A.5.3.1 Real life use cases 232
A.5.3.2 Loop-free Repair Mechanisms 232
A.5.3.3 Heterogeneity 233
A.5.3.4 DIS handling 233
A.5.4 Research challenges 233
A.5.4.1 Interaction with MAC protocols 233
A.5.4.2 Asymmetric links 233
A.5.4.3 Mobility 234
A.5.4.4 Multi-Sink support 235
A.5.4.5 Scalability of the Non-Storing RPL approach . . 235
A.6 IETF CoRE Working Group 235
A.6.1 Key protocols 235
A.6.1.1 Base CoAP 235
A.6.1.2 CoRE Link Format 238
A.6.1.3 Block transfer 239
A.6.1.4 Observation of resource 240
A.6.2 Implementation and evaluation 240
A.6.2.1 CoAP implementations 240
A.6.2.2 CoAP performance evaluation 241
A.6.3 Leveraging upon CoAP to realize the IoT 243
A.6.3.1 Discovery and Naming 243
A.6.3.2 Congestion control 244
A.6.3.3 Advanced interaction patterns 244
A.6.3.4 Communication with Sleepy Nodes 245
A.6.3.5 Security 246
A.6.3.6 Intermediaries 246
A.6.3.7 CoAP in cellular networks 246

A.6.3.8 Real life use cases of CoAP in the IoT 247
A.6.4 Research challenges 248
A.7 Using IETF standards to realize the Internet of Things 250
A.7.1 Overview of the IETF LLN Protocol Stack 250
A.7.2 Realizing the Web of Things 250
A.7.3 Interoperability 251
A.7.4 Bringing semantics to the Web of Things 252
A.7.5 Security and privacy in the Web of Things 253
A.7.6 Reprogrammability 254
A.8 Conclusions 254
References 256

B Internet of Things Virtual Networks: Bringing Network Virtualization to Resource-constrained Devices 271
B.1 Introduction 272
B.2 Current approaches and limitations 273
B.2.1 Use of gateways 274
B.2.2 Integration of sensors into the IP-world 275
B.3 IoT-VN 276
B.4 Example use cases 278
B.4.1 Partitioning a sensor network 278
B.4.2 Aggregating multiple sensor networks 279
B.4.3 Extending a sensor network with non-constrained devices 280
B.4.4 A hybrid of sensor networks 280
B.5 Implementation 281
B.5.1 Non-constrained implementation 281
B.5.2 Extension of the IoT-VN concept to resource-constrained devices 282
B.5.3 Protocols inside the IoT-VN 283
B.5.3.1 AODV 283
B.5.3.2 Ping 284
B.6 Results 284
B.7 Conclusions and outlook 288
References 289

List of Figures

1.1 The expansion of the Internet . . . 2
1.2 Google Trends for the Internet of Things (2004 – 2015) . . . 3
1.3 Sample IoT application domains. . . . 4
1.4 Sample IoT devices. . . . 6
1.5 Sample sensor nodes (motes). . . . 7
1.6 Constrained Environments architecture . . . 8
1.7 Gateways and proprietary protocols are often used to interconnect sensor networks to the Internet . . . 9
1.8 Internet protocols are extended to the sensor networks . . . 11
1.9 IETF Low-power and Lossy Network (LLN) protocol stack . . . 11
1.10 Schematic position of the different chapters in this dissertation . . . 14

2.1 Constrained Environments architecture and the CoAP protocol stack 28
2.2 An example of resource discovery and access using CoAP . . . 29
2.3 Network topology used to present the self-organization solution . 30
2.4 Complete self-organization process . . . 32
2.5 A step-by-step example illustrating the browsable hierarchy . . . 40
2.6 A sample of a direct end-to-end access to a CoAP resource . . . 41
2.7 Name resolution using dynamically configured DNS information . 42
2.8 Effect of introducing random delays for two different network sizes 44

3.1 CoAP Message Format . . . 61
3.2 An example of a direct resource discovery and a CoAP request. . . 64
3.3 An example resource discovery by using a Resource Directory . . 65
3.4 System components overview . . . 72
3.5 Possible locations of the Entity Manager . . . 73
3.6 Entity Manager high-level structure . . . 74
3.7 Entity creation example . . . 75
3.8 Entity validation process flow . . . 77
3.9 Simplified entity usage process flow . . . 79
3.10 Example entity usage . . . 80
3.11 iMinds IoT portable home automation demo box. . . . 83
3.12 Screenshot: Entity creation . . . 84
3.13 Screenshot: Entity usage . . . 85
3.14 Advanced Entity Manager (EM) features. . . . 86

3.15 Screenshot: Entity creation using a resource directory 87
3.16 Network topology used in performance evaluation experiments . . 88
3.17 Effect of adding delays between requests on the response time . . 91
3.18 The reliability of individual group members 93
3.19 The reliability of the complete group 94
3.20 Number of packets sent vs. packet loss 97
3.21 Average group response time 98
3.22 Scattergram of the response times 99
3.23 Number of packets while using a cache 100
3.24 Response time with and without a cache 101
3.25 Number of packets needed to validate a single entity member . . . 102
3.26 Average entity validation time 103

4.1 CoAP direct resource discovery 120
4.2 Observing a resource in CoAP 121
4.3 Overview of involved components 124
4.4 Potential locations of the Entity Manager 124
4.5 High-Level structure of the EM 125
4.6 Entity creation example 127
4.7 Handling client requests to use existing entities 127
4.8 Example entity usage 128
4.9 Observe notifications 130
4.10 The ratio between client and member notifications when using the `max` Entity Operation 133
4.11 Reducing the precision of notifications 134
4.12 Ratio between client and member notifications and the average delay for member notifications as a function of the Notifications Aggregation Window. 135
4.13 Notifications Aggregation Window 137
4.14 Location of the Entity Manager within the CoAP++ architecture . 138
4.15 Screen-shots using the CoAP++ client GUI to create, observe and delete an observable entity of three members. 140
4.16 The ratio between the number of client notifications to the number of member notifications for different Entity Operations 142
4.17 Reducing the precision of the entity 143
4.18 The effect of the Notifications Aggregation Window size on the number of client notifications and the ratio between client and member notifications 145
4.19 The effects of the change window and the group size on the ratio between client and member notifications for various Notifications Aggregation Window sizes 146
4.20 Combining entity properties achieves even better optimization. . . 148

5.1 Example of CoAP Non-confirmable Message (NON) exchange . . 163
5.2 Example of CoAP Confirmable Message (CON) exchange 164

5.3 Screenshots using the CoAP++ client GUI to create, query and delete a multicast entity of three members. 173
5.4 Experimental setup at w-iLab.t Zwijnaarde Testbed 176
5.5 Entity response time vs. delay between requests 178
5.6 Entity response time vs. delay between requests for different group sizes 179
5.7 Entity response time per member vs. delay between requests for different group sizes 180
5.8 The reliability of individual group members 181
5.9 The reliability of the complete group 183
5.10 Average group response time 184
5.11 Entity response time vs. the speed of interfering Wi-Fi traffic . . . 185
5.12 Effect of the Wi-Fi interference on the response time for different values of the initial back-off time 187
5.13 Experimental setup at w-iLab.t office 188
5.14 Response times for unicast and multicast on w-iLab.t office 189

A.1 Gateways and proprietary protocols are often used to interconnect sensor networks to the Internet 206
A.2 Internet protocols are extended to the sensor networks 207
A.3 IEEE 802.15.4 PPDU and MPDU formats 209
A.4 6LoWPAN encapsulation header stack examples 213
A.5 IPHC Compression 214
A.6 Internet Protocol Version 6 (IPv6) Header Compression Example . 216
A.7 6LoWPAN Fragmentation Header 218
A.8 6LoWPAN Mesh Addressing Header 219
A.9 Packet flow for point-to-point traffic between two nodes in an RPL network 226
A.10 ICMPv6 header for RPL control messages 226
A.11 The DODAG Information Object (DIO) Base Object 227
A.12 The DAO Base Object 228
A.13 CoAP Message Format 236
A.14 CoAP Option Format 237
A.15 An example of CoRE resource discovery and CoAP request 239
A.16 IETF LLN protocol stack 250

B.1 Gateways are often used to interconnect sensor networks to the Internet 274
B.2 Internet protocols are extended to the sensor networks 276
B.3 MENO Concept – IoT-VN approach 277
B.4 An IoT-VN that only contains a subset of the available sensors . . 279
B.5 An IoT-VN that combines several sensor networks 279
B.6 An IoT-VN that is extended to include non-constrained devices . . 280
B.7 A hybrid IoT-VN 280
B.8 An IoT-VN that contains layer 2 and layer 3 virtual links 282

B.9 IoT-VN Network header and ping application data format 283
B.10 The network that was used to test the IoT-VN implementation . . 284
B.11 Screenshot from the ping application showing successful ping between two sensor that belong to the same IoT-VN 285
B.12 A ping packet traveling over a virtual network 287

List of Tables

1.1 Classes of Constrained Devices 6
1.2 An overview of the contributions per chapter in this dissertation. . 16

2.1 Summary of the experiments showing the effect of changing the maximum random delays in forwarding broadcasts and in responding to CoAP discovery broadcasts (Leisure) on the discovery rate and delay for two different test networks 45

3.1 Cooja network simulator settings. 88
3.2 Summary of communication delays in used topology. 92

4.1 Contiki and Cooja network simulator settings 141

5.1 When multicast is enabled on Contiki nodes, they automatically join four Link-Local groups. 170
5.2 The group membership management resource `/mc` interface. . . . 170
5.3 Resource availability and requirements 171
5.4 Evaluation experiments settings. 177
5.5 Characteristics of the Wireless Sensor Networks (WSNs) used in congestion control experiments 179
5.6 Summary of the results of the experiments on w-iLab.t office wireless sensor testbed. 188

A.1 Summary of Dispatch Byte values. 213
A.2 Summary of address compression fields. 215
A.3 6LoWPAN implementations. 220
A.4 Code fields in RPL ICMPv6 messages. 227
A.5 Implementations incorporating RPL. 228
A.6 Simulators incorporating RPL. 229
A.7 CoAP implementations 242

B.1 Ping round trip times in millisecond. 286

List of Acronyms

0-9

6LoWPAN IPv6 over Low power Wireless Personal Area Networks
6TiSCH IPv6 over the TSCH mode of IEEE 802.15.4e

A

ACK Acknowledgment
AODV Ad hoc On-Demand Distance Vector

B

BAS Building Automation System
BR Border Router

C

CoAP Constrained Application Protocol
CON Confirmable Message
CoRE Constrained RESTful Environments
CPU Central Processing Unit
CSMA Carrier Sense multiple Access

D

DAG Directed Acyclic Graph

DHCP Dynamic Host Configuration Protocol
DIY Do-It-Yourself
DNS Domain Name System
DNS-SD DNS Service Discovery
DODAG Destination-Oriented DAG
DTLS Datagram Transport Layer Security

E

EM Entity Manager
ETSI European Telecommunications Standards Institute

F

FQDN Fully Qualified Domain Name

G

GSM Global System for Mobile Communications
GUI Graphical User Interface
GW Gateway

H

HTML HyperText Markup Language
HTTP Hypertext Transfer Protocol
HVAC Heating, Ventilating, and Air Conditioning

I

IEEE Institute of Electrical and Electronics Engineers
IESG Internet Engineering Steering Group

IETF Internet Engineering Task Force
IoT Internet of Things
IoT-VN Internet of Things Virtual Network
IP Internet Protocol
IPv6 Internet Protocol version 6
IPSO Internet Protocol for Smart Objects
ISM Industrial, Scientific and Medical

K

KB KiloByte
kbps kilobit per second

L

LAN Local Area Network
LLN Low-power and Lossy Network
LPL Low Power Listening
LWM2M Lightweight M2M

M

M2M Machine-to-Machine
MAC Media Access Control
mDNS multicast Domain Name System
mDNS-SD Multicast DNS - Service Discovery
MENO Managed Ecosystems of Networked Objects
MID Message ID
MPDU MAC Protocol Data Unit
MPL Multicast Protocol for Low power and Lossy Networks
MQTT Message Queuing Telemetry Transport
MTU Maximum Transmission Unit

N

NON Non-confirmable Message

O

OMA Open Mobile Alliance

P

PC Personal Computer
PPDU Physical Protocol Data Unit
PSDU Physical Service Data Unit

Q

QoS Quality of Service

R

RAM Random Access Memory
RD Resource Directory
RDC Radio Duty Cycle
RDF Resource Description Framework
REST Representational State Transfer
RF Radio Frequency
RFC Request For Comment
ROLL Routing Over Low power and Lossy networks
ROM Read Only Memory
RPL IPv6 Routing Protocol for Low-Power and Lossy Networks
RST Reset Message

S

SMRF Stateless Multicast RPL Forwarding

T

TCP Transmission Control Protocol
TDMA Time Division Multiple Access
TM Trickle Multicast
TSCH Timeslotted Channel Hopping

U

UDP User Datagram Protocol
URI Uniform Resource Identifier

V

VLAN Virtual Local Area Network
VPN Virtual Private Network

W

WAN Wide Area Network
WBAN Wireless Body Area Network
Wi-Fi Wireless Fidelity
WoT Web of Things
WSAN Wireless Sensor and Actuator Network
WSN Wireless Sensor Network
WWW World Wide Web

Samenvatting
– Summary in Dutch –

Het alomtegenwoordige Internet is snel aan het uitbreiden naar nieuwe domeinen. De snelheid waarmee het Internet uitbreidt, is te vergelijken met de verspreiding van het Internet in de jaren negentig. Dit uitgebreide Internet wordt tegenwoordig gewoonlijk het *Internet of Things* (IoT) genoemd en men verwacht dat het ongeveer 50 miljard toestellen met elkaar zal verbinden tegen het jaar 2020. Dat betekent dat binnen slechts 5 jaar na het schrijven van dit doctoraat het aantal verbonden toestellen het aantal mensen zeven maal zal overschrijden. Verder wordt er verwacht dat het overgrote deel van deze IoT toestellen ingebedde toestellen met gelimiteerde mogelijkheden zullen zijn, zoals sensoren en actuatoren. Sensoren verzamelen informatie over de fysieke wereld en injecteren deze informatie in de virtuele wereld. Vervolgens kan deze informatie worden aangewend om acties te ondernemen in de echte wereld door middel van actuatoren die een bepaalde taak uitvoeren.

De integratie van ingebedde toestellen in het Internet introduceert nieuwe uitdagingen. Dit komt doordat vele van de huidige Internet technologieën en protocollen niet zijn ontworpen voor dergelijke gelimiteerde toestellen. Omdat deze toestellen zijn geoptimaliseerd voor lage productiekosten en een laag energieverbruik, hebben ze beperkingen inzake energievoorziening, geheugen en rekencapaciteit. Bovendien slapen ze gedurende lange periodes. Ook de netwerken gevormd door deze ingebedde toestellen zijn gelimiteerd en hebben karakteristieken die afwijken van wat gangbaar is in het hedendaagse Internet. Deze gelimiteerde netwerken hebben veel pakketverlies, een lage doorvoersnelheid, frequente veranderingen in topologie en een beperkte pakketgrootte. Netwerken die weinig stroom verbruiken en pakketverlies vertonen worden *Low power and Lossy Networks* (LLN) genoemd. Door hun bijzondere kenmerken is het in de meeste gevallen onhaalbaar om te communiceren via conventionele Internet protocollen met gelimiteerde toestellen en LLNs. Nieuwe of aangepaste protocollen die deze bijzondere eigenschappen in acht nemen, zijn noodzakelijk.

In de voorbije jaren zijn er veel inspanningen geleverd om de uitbreiding van Internet technologieën naar gelimiteerde toestellen mogelijk te maken. Initieel werd de nadruk gelegd op de netwerklaag. De uitbreiding van het Internet in de jaren negentig kwam er echter niet alleen door de introductie van nieuwe of betere netwerkprotocollen, maar wel door het introduceren van het *World Wide Web* (WWW), welke het integreren van diensten en applicaties eenvoudiger maakte.

Een van de essentiële technologieën die aan de basis van het WWW lag was het *Hypertext Transfer Protocol* (HTTP). Vandaag is HTTP een van de belangrijkste protocollen voor de realisatie van schaalbare internetdiensten gebaseerd op het *Representational State Transfer* (REST) paradigma. De REST architectuur maakt de realisatie van schaalbare en performante diensten mogelijk, gebruikmakend van uniforme en eenvoudige interfaces. Indien er een HTTP protocol geschikt voor ingebedde systemen voorhanden zou zijn, dan zou dit de opkomst van het IoT ten goede kunnen komen.

Recent is bijgevolg het werk begonnen dat de integratie van gelimiteerde toestellen in het Internet moet faciliteren op serviceniveau. De *Internet Engineering Task Force* (IETF) *Constrained RESTful Environments* (CoRE) werkgroep heeft een REST architectuur voor de meest gelimiteerde toestellen en netwerken gedefinieerd. Daarvoor werd het *Constrained Application Protocol* (CoAP) geïntroduceerd: een RESTful web transfer protocol gespecialiseerd voor gebruik met gelimiteerde netwerken en toestellen. CoAP realiseert een deel van de REST mechanismen uit HTTP, maar is geoptimaliseerd voor *Machine-2-Machine* (M2M) toepassingen.

Dit doctoraatsonderzoek bouwt verder op CoAP om een betere integratie van gelimiteerde toestellen in het IoT mogelijk te maken en onderzoekt voorgestelde CoAP oplossingen, zowel theoretisch als experimenteel, en stelt alternatieven voor waar nodig. Het eerste deel van dit onderzoek stelt een mechanisme voor dat het uitrollen van sensornetwerken vereenvoudigt en de ontdekking, end-to-end connectiviteit en dienstengebruik van nieuw uitgerolde sensoren mogelijk maakt. De voorgestelde aanpak maakt gebruik van CoAP en combineert het met het *Domain Naming System* (DNS) om sensoren te adressen door middel van gebruiksvriendelijke *Fully Qualified Domain Names* (FQDNs). Dit omvat de automatische ontdekking van sensoren en sensor gateways en de vertaling van HTTP naar CoAP. Hierdoor maakt de gevolgde aanpak sensordiensten globaal kenbaar en bereikbaar van eender welke Internet-geconnecteerde client door middel van IPv6 adressen of DNS namen via zowel HTTP als CoAP. Het voorstel voorziet aldus een haalbare en flexibele oplossing om hiërarchische zelforganisatie tot stand te brengen met een minimum aan pre-configuratie. Dit laat ons toe om we dure menselijke interventies tot een minimum tot beperken en om de introductie van nieuwe protocollen voor het ontdekken en het beheren van sensordiensten te vermijden. Dit reduceert zowel de kost als de grootte van de software implementatie van de gelimiteerde toestellen.

Het tweede en grotere deel van dit onderzoek focust op het gebruik van CoAP om communicatie tot stand te brengen met groepen van sensoren en hun diensten. In veel IoT applicatiedomeinen moeten sensoren of actuatoren geadresseerd worden als groep in plaats van individueel, aangezien individuele middelen vaak niet toereikend of nuttig kunnen zijn. Een simpel voorbeeld is een kamer waar alle lichten in een kamer samen aan of uit moeten gaan wanneer iemand op een lichtschakelaar drukt. Aangezien misschien niet alle IoT toepassingen groepscommunicatie nodig hebben, heeft de CoRE werkgroep deze niet in de basis CoAP specificatie opgenomen. Hierdoor blijft de basis specificatie zo efficiënt en simpel

mogelijk, zodat het kan gebruikt worden op zelfs de meest gelimiteerde toestellen. Groepscommunicatie en andere componenten die optioneel zijn voor sommige toestellen, worden gestandaardiseerd als aparte uitbreidingen. Eerst hebben we de voorgestelde uitbreiding voor groepscommunicatie onderzocht, welke gebruik maakt van *Internet Protocol version 6* (IPv6) multicast communicatie. We wijzen op de belangrijkste sterktes en zwaktes en stellen onze eigen complementaire oplossing voor die unicast communicatie gebruikt om groepscommunicatie te verwezenlijken. Onze oplossing biedt mogelijkheden die verder gaan dan simpele groepscommunicatie. We voorzien bijvoorbeeld een validatiemechanisme dat verschillende controles uitvoert op de groepsleden, om er zeker van te zijn dat het mogelijk is om ze combineren. We maken het ook mogelijk voor de client om te vragen dat de resultaten van de individuele leden eerst verwerkt worden vooraleer ze naar de client verstuurd worden. De client kan bijvoorbeeld vragen om alleen de maximum waarde van alle individuele leden te sturen.

Een volgende belangrijke optionele uitbreiding van CoAP maakt het mogelijk voor clients om voortdurend de toestand van sensoren en actuatoren te observeren. Hiertoe dienen ze hun interesse te registreren in het ontvangen van notificaties van CoAP servers telkens deze toestand verandert. Door dit publiceer/registreer mechanisme te gebruiken hoeft de client niet voortdurend de toestellen bevragen om te bepalen of een waarde al dan niet veranderd is. Dit leidt doorgaans tot efficiëntere communicatiepatronen en een lagere belasting van toestellen en LLNs. Helaas is deze CoAP functionaliteit voor observaties niet compatibel met de CoAP uitbreiding voor groepscommunicatie, aangezien de uitbreiding voor observaties unicast communicatie veronderstelt, terwijl de groepscommunicatie uitbreiding enkel multicast communicatie ondersteunt. Deze thesis stelt voor om onze eigen oplossing voor groepscommunicatie uit te breiden met mogelijkheden voor groepsobservatie. Door groepsobservatie te combineren met groepsverwerkingsfunctionaliteit, wordt het mogelijk om een client enkel in te lichten over de veranderingen binnen de geobserveerde groep (bijvoorbeeld de maximumwaarde van alle groepsleden is veranderd).

Aangezien zowel het gebruik van multicast als van unicast voor groepscommunicatie sterktes als zwaktes heeft, stellen wij voor om onze unicast gebaseerde oplossing uit te breiden met multicast capaciteiten. Op deze wijze proberen we de sterktes van beide benaderingen te combineren om over het algemeen een betere groepscommunicatie te verwezenlijken, die bovendien flexibel is en die kan aangepast worden naargelang de noden van specifieke toepassingsscenarios.

Samen vormen de voorgestelde mechanismen een krachtige en complete oplossing voor het uitdagende probleem van groepscommunicatie met ingebedde toestellen. We hebben de voorgestelde oplossingen uitgebreid en op verschillende vlakken geëvalueerd. Waar mogelijk werd er een theoretisch model opgesteld dat werd gevalideerd door middel van numerieke simulaties. We hebben deze oplossingen ook experimenteel geëvalueerd en vergeleken met andere voorgestelde oplossingen gebruikmakend van een kleine demonstratie-opstelling en later ook van twee grote en schaalbare sensor testbeds en dat alles onder verschillende testcondities. Het eerste testbed bevindt zich in een grote afgeschermde ruimte, wat

testen onder gecontroleerde condities mogelijk maakt. Het tweede testbed bevindt zich in een operationeel kantoorgebouw, wat testen onder normale condities mogelijk maakt. Deze testen onthulden onregelmatigheden inzake performantie en andere echte problemen. We stellen enkele oplossingen en suggesties voor om deze problemen aan te pakken.

Afgezien van de hoofdcontributies, zijn er twee andere relevante resultaten van dit doctoraatsonderzoek opgenomen als appendices. De eerste appendix omvat een samenvatting van de belangrijkste standaardisatie-activiteiten binnen het IETF met betrekking tot het IoT. Verder tonen we ook aan dat met de introductie van CoAP er een volledige stack van gestandaardiseerde protocollen voorhanden is die de volledige communicatie stack afdekt en aldus de stap naar het *Web of Things* (WoT) mogelijk maakt. Door enkel gestandaardiseerde protocollen te gebruiken, kunnen toestellen van verschillende producenten in 1 groot WoT geïntegreerd worden waar deze bruikbaar zijn voor zowel mensen als machines.

De tweede appendix beschrijft een alternatieve oplossing voor het groeperen van gelimiteerde toestellen door middel van virtualisatietechnieken. Onze aanpak focust op objecten, zowel gelimiteerde als ongelimiteerde toestellen, die moeten samenwerken en integreert deze in een beveiligd virtueel netwerk. Dergelijke netwerken noemen we *Internet of Things Virtual Networks* of IoT-VNs. Binnen zo een IoT-VN wordt end-to-end communicatie mogelijk gemaakt via protocollen welke de beperkingen van gelimiteerde toestellen in acht nemen. Verder beschrijft deze appendix hoe het IoT-VN concept kan toegepast worden in een aantal generieke toepassingen en hoe het aldus een volwaardig alternatief kan vormen ter ondersteuning van specifieke toepassingen.

Summary

The ubiquitous Internet is rapidly spreading to new domains. This expansion of the Internet is comparable in scale to the spread of the Internet in the '90s. The resulting Internet is now commonly referred to as the Internet of Things (IoT) and is expected to connect about 50 billion devices by the year 2020. This means that in just five years from the time of writing this PhD the number of interconnected devices will exceed the number of humans by sevenfold. It is further expected that the majority of these IoT devices will be resource constrained embedded devices such as sensors and actuators. Sensors collect information about the physical world and inject this information into the virtual world. Next processing and reasoning can occur and decisions can be taken to enact upon the physical world by injecting feedback to actuators.

The integration of embedded devices into the Internet introduces new challenges, since many of the existing Internet technologies and protocols were not designed for this class of constrained devices. These devices are typically optimized for low cost and power consumption and thus have very limited power, memory, and processing resources and have long sleep periods. The networks formed by these embedded devices are also constrained and have different characteristics than those typical in todays Internet. These constrained networks have high packet loss, low throughput, frequent topology changes and small useful payload sizes. They are referred to as LLN. Therefore, it is in most cases unfeasible to run standard Internet protocols on this class of constrained devices and/or LLNs. New or adapted protocols that take into consideration the capabilities of the constrained devices and the characteristics of LLNs, are required.

In the past few years, there were many efforts to enable the extension of the Internet technologies to constrained devices. Initially, most of these efforts were focusing on the networking layer. However, the expansion of the Internet in the 90s was not due to introducing new or better networking protocols. It was a result of introducing the World Wide Web (WWW), which made it easy to integrate services and applications. One of the essential technologies underpinning the WWW was the Hypertext Transfer Protocol (HTTP). Today, HTTP has become a key protocol in the realization of scalable web services building around the Representational State Transfer (REST) paradigm. The REST architectural style enables the realization of scalable and well-performing services using uniform and simple interfaces. The availability of an embedded counterpart of HTTP and the REST architecture could boost the uptake of the IoT.

Therefore, more recently, work started to allow the integration of constrained

devices in the Internet at the service level. The Internet Engineering Task Force (IETF) Constrained RESTful Environments (CoRE) working group has realized the REST architecture in a suitable form for the most constrained nodes and networks. To that end the Constrained Application Protocol (CoAP) was introduced, a specialized RESTful web transfer protocol for use with constrained networks and nodes. CoAP realizes a subset of the REST mechanisms offered by HTTP, but is optimized for Machine-to-Machine (M2M) applications.

This PhD research builds upon CoAP to enable a better integration of constrained devices in the IoT and examines proposed CoAP solutions theoretically and experimentally proposing alternatives when appropriate. The first part of this PhD proposes a mechanism that facilitates the deployment of sensor networks and enables the discovery, end-to-end connectivity and service usage of newly deployed sensor nodes. The proposed approach makes use of CoAP and combines it with Domain Name System (DNS) in order to enable the use of user-friendly Fully Qualified Domain Names (FQDNs) for addressing sensor nodes. It includes the automatic discovery of sensors and sensor gateways and the translation of HTTP to CoAP, thus making the sensor resources globally discoverable and accessible from any Internet-connected client using either IPv6 addresses or DNS names both via HTTP or CoAP. As such, the proposed approach provides a feasible and flexible solution to achieve hierarchical self-organization with a minimum of pre-configuration. By doing so we minimize costly human interventions and eliminate the need for introducing new protocols dedicated for the discovery and organization of resources. This reduces both cost and the implementation footprint on the constrained devices.

The second, larger, part of this PhD focuses on using CoAP to realize communication with groups of resources. In many IoT application domains, sensors or actuators need to be addressed as groups rather than individually, since individual resources might not be sufficient or useful. A simple example is that all lights in a room should go on or off as a result of the user toggling the light switch. As not all IoT applications may need group communication, the CoRE working group did not include it in the base CoAP specification. This way the base protocol is kept as efficient and as simple as possible so it would run on even the most constrained devices. Group communication and other features that might not be needed by all devices are standardized in a set of optional separate extensions. We first examined the proposed CoAP extension for group communication, which utilizes Internet Protocol version 6 (IPv6) multicasts. We highlight its strengths and weaknesses and propose our own complementary solution that uses unicast to realize group communication. Our solution offers capabilities beyond simple group communication. For example, we provide a validation mechanism that performs several checks on the group members, to make sure that combining them together is possible. We also allow the client to request that results of the individual members are processed before they are sent to the client. For example, the client can request to obtain only the maximum value of all individual members.

Another important optional extension to CoAP allows clients to continuously observe resources by registering their interest in receiving notifications from CoAP

servers once there are changes to the values of the observed resources. By using this publish/subscribe mechanism the client does not need to continuously poll the resource to find out whether it has changed its value. This typically leads to more efficient communication patterns that preserve valuable device and LLN resources. Unfortunately CoAP observe does not work together with the CoAP group communication extension, since the observe extension assumes unicast communication while the group communication extension only support multicast communication. In this PhD we propose to extend our own group communication solution to offer group observation capabilities. By combining group observation with group processing features, it becomes possible to notify the client only about certain changes to the observed group (e.g., the maximum value of all group members has changed).

Acknowledging that the use of multicast as well as unicast has strengths and weaknesses we propose to extend our unicast based solution with certain multicast features. By doing so we try to combine the strengths of both approaches to obtain a better overall group communication that is flexible and that can be tailored according to the use case needs.

Together, the proposed mechanisms represent a powerful and comprehensive solution to the challenging problem of group communication with constrained devices. We have evaluated the solutions proposed in this PhD extensively and in a variety of forms. Where possible, we have derived theoretical models and have conducted numerous simulations to validate them. We have also experimentally evaluated those solutions and compared them with other proposed solutions using a small demo box and later on two large scale wireless sensor testbeds and under different test conditions. The first testbed is located in a large, shielded room, which allows testing under controlled environments. The second testbed is located inside an operational office building and thus allows testing under normal operation conditions. Those tests revealed performance issues and some other problems. We have provided some solutions and suggestions for tackling those problems.

Apart from the main contributions, two other relevant outcomes of this PhD are described in the appendices. In the first appendix we review the most important IETF standardization efforts related to the IoT and show that with the introduction of CoAP a complete set of standard protocols has become available to cover the complete networking stack and thus making the step from the IoT into the Web of Things (WoT). Using only standard protocols makes it possible to integrate devices from various vendors into one big WoT accessible to humans and machines alike.

In the second appendix, we provide an alternative solution for grouping constrained devices by using virtualization techniques. Our approach focuses on the objects, both resource-constrained and non-constrained, that need to cooperate by integrating them into a secured virtual network, named an Internet of Things Virtual Network or IoT-VN. Inside this IoT-VN full end-to-end communication can take place through the use of protocols that take the limitations of the most resource-constrained devices into account. We describe how this concept maps to several generic use cases and, as such, can constitute a valid alternative approach

for supporting selected applications.

1 Introduction

"Say not, 'I have found the truth,' but rather, 'I have found a truth.'"

– Kahlil Gibran (1883 - 1931)

This chapter situates the conducted research work, summarizes the main contributions and outlines the structure of this dissertation. It also provides an overview of the publications that were authored during this research period.

1.1 Context

The ubiquitous Internet is rapidly spreading to new domains. This expansion of the Internet is comparable in scale to the spread of the Internet in the '90s. The resulting Internet is now commonly referred to as the Internet of Things (IoT) to reflect the fact that it is connecting many *Things*, i.e., not just people, computers and mobile phones as it is common on the Internet nowadays. With an adaption rate that is five times faster than that of the rate of adaption for electricity or telephony, the IoT is expected to connect about 50 billion devices by the year 2020 (Figure 1.1) [1]. This means that in just five years from the time of writing this dissertation the number of interconnected devices will exceed the number of humans by sevenfold.

As one might suspect, connecting this huge number of devices brings many challenges that have to be overcome on the way. These challenges are not just related to the amount of devices to be connected, but also among others, to the wide

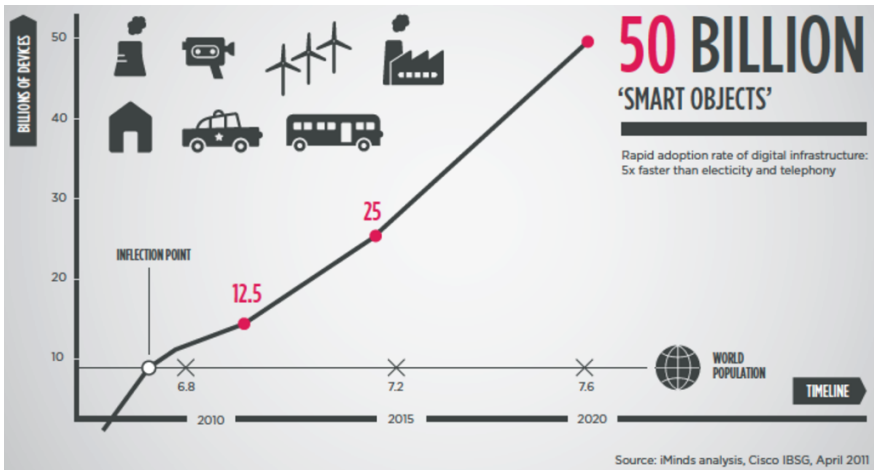


Figure 1.1: The number of devices that are connected to the Internet exceeds the number of humans since a few years. It is expected that by the year 2020 about 50 billion devices will be connected to the Internet [1].

range of requirements and capabilities of these devices. One of the main challenges is to have a common set of open standards and enablers for the development of a new category of applications that enables the users to benefit from connecting this huge amount of things to the Internet. This is the main context for this dissertation, which is further elaborated in the following two subsections on the IoT and open standards.

1.1.1 The Internet of Things

There are many definitions for the IoT that vary in complexity and detail. Some definitions consider the IoT to refer to an infrastructure while other definitions consider it to be just a concept (or a phenomenon), without referring to a network infrastructure [2]. In order to avoid introducing yet another definition for the IoT, we use in this dissertation the simple definition from [3]:

> **The Internet of Things** is a system of physical objects that can be discovered, monitored, controlled or interacted with by electronic devices which communicate over various networking interfaces, and eventually can be connected to the wider Internet.

Although the term IoT has been around since 1999 [4] and has been used a lot in the research community, it has not been receiving a lot of public attention. However, this seems to have changed over the last two years as can be seen from Google Trends when looking for the "IoT" field of study (Figure 1.2). This ex-

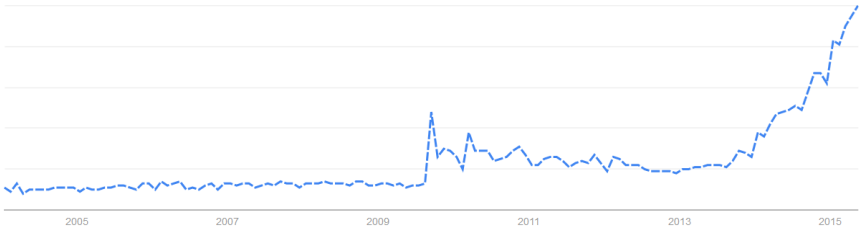


Figure 1.2: Google Trends when looking for the Internet of Things field of study (2004 – 2015). After several years of almost constant public interest, the last two years show an exponential trend in the interest.

pansion of the public interest in the IoT in the past two years is a reflection of the introduction of new IoT application domains and the increase of available IoT devices and device types. In the remainder of this subsection we briefly provide some sample IoT application domains, IoT devices, and IoT networks.

1.1.1.1 IoT application domains

The IoT is rapidly expanding in many application domains (Figure 1.3). Some of these domains are traditionally connected domains, but many other domains were traditionally not connected and are gradually discovering the benefits of interconnectivity. Following are some sample IoT application domains along with a few sample usages.

Building automation/Smart home is the automatic centralized control of a building's Heating, Ventilating, and Air Conditioning (HVAC), lighting, window blinds and other systems through a Building Automation System (BAS). The goals of building automation are to improve the occupants' comfort and at the same time to reduce energy consumption and operating costs through an efficient operation of building systems.

Smart cities use digital technologies to enhance the performance of the city sectors such as transport, energy, health care, water and waste management. The goals are to improve the wellbeing of the citizens and to engage more effectively and actively with them to reduce costs and resource consumption. It is believed that smart cities will be able to respond faster to challenges than traditional cities.

Health IoT applications are very diverse. One important example is the monitoring of the personal health and fitness by using various sensors typically in the form of wearable devices. These devices can monitor personal activity (e.g., number of steps walked, calories

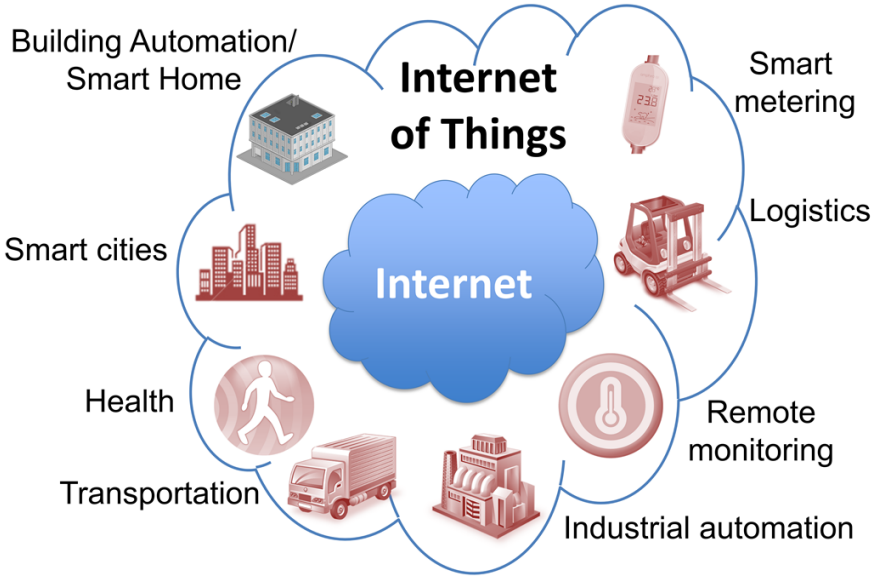


Figure 1.3: Sample IoT application domains.

burned, *etc*.), vitals (heart beat, *etc*.) and data of more medical relevance such as the blood glucose level. These devices communicate by forming a Wireless Body Area Network (WBAN) and their data can be accessed by smart phones, watches, computers, *etc*. for further analysis.

Transportation of persons and goods is an important application domain for the IoT, since it is a field that can be optimized to save energy and cost. In 2013, shippers moved more than eight billion tons of cargo around the world [1]. The ability to precisely monitor and track containers is essential for 'just-in-time' delivery, the realization of safe trade lanes and can effectively reduce costs.

Smart metering usually refers to the use of electronic devices to record consumption of various resources (such as electricity, gas and water) and to communicates that information, at least daily, back to the utility for monitoring and billing. Smart meters enable two-way communication between the meter and the central system and thus can be used for example to disconnect-reconnect the service remotely and thus help to reduce costs. There are 1 billion electricity meters in the world [5] and maybe a similar number of water and other utility meters that will sooner or later be converted to smart meters.

Smart logistics was the field in which the term Internet of Things was first used

[4] and thus has the longest history as an IoT application domain. By tagging any kind of product with a unique identifier that can be read by electronic devices, this product becomes part of the IoT. This product can then be tracked from the moment the tag is added (possibly at manufacturing time) until it is removed by the end user or reaches its end of life at a recycling park. This helps to reduce cost of re-tagging the product with temporary tags for shipping and retail purposes. Further, it can also contribute to a reduction in counterfeiting.

Remote monitoring was initially limited to SCADA (supervisory control and data acquisition) technology and referred to the measurement of disparate devices from a network operations center and the ability to change the operation of these devices from that central place. Remote monitoring and control is entering a new era with the development of wireless sensors and the IoT. It is being used in many fields such as patient monitoring, smart grid, pipeline sensors, and desktop/server monitoring.

Industrial automation is the use of various control systems for operating equipment with minimal or reduced human intervention. It is believed that this domain is being revolutionized by the IoT [3]. Manufacturing plants are being transformed into flexible eco-systems of reconfigurable production lines that can rearrange themselves to perform any given tasks as efficiently as possible. The IoT is a main enabler of this transformation since it enables industrial machines to offer services, making it possible to use and reuse them in combination with other machines by connecting their respective services.

1.1.1.2 IoT things

Looking at the wide range of application domains and their connectivity needs, it becomes obvious that also a wider range of objects will be connected to the IoT to fulfill these needs. The IoT refers to the network of networks of everyday objects, i.e., a network of uniquely addressable interconnected objects. The term *Things* in the IoT refers to very various objects such as people, devices, clothes, food, books, dogs, *etc*. (Figure 1.4). These devices have different capabilities and have different communication requirements. Unlike today's Internet, it is expected that the majority of the IoT devices will be resource constrained embedded devices such as sensors and actuators. Sensors collect information about the physical world and inject this information into the virtual world. Next processing and reasoning can occur and decisions can be taken to enact upon the physical world by injecting feedback to actuators.

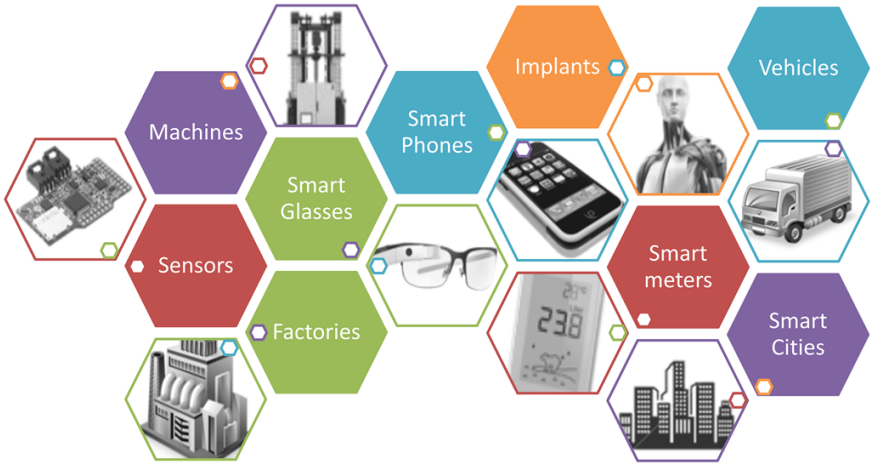


Figure 1.4: Sample IoT devices.

Sensors, actuators and other embedded devices are often optimized for low cost and power consumption and thus have very limited power, memory, and processing resources and have long sleep periods. These resource constrained devices, or just constrained devices for the sake of simplicity, are classified in [6] into three classes according to their memory resources (Table 1.1):

Table 1.1: Classes of Constrained Devices [6]

Name	**Data Size** (e.g., RAM)	**Code Size** (e.g., Flash)
Class 0	$<<$ 10 KB	$<<$ 100 KB
Class 1	$\approx$ 10 KB	$\approx$ 100 KB
Class 2	$\approx$ 50 KB	$\approx$ 250 KB

Class 0 devices are so severely constrained in their resources that most likely they will not be able to communicate directly with the Internet in a secure manner. They will participate in Internet communications with the help of larger devices such as proxies or gateways.

Class 1 devices are quite constrained in their resources, such that they cannot easily employ a full Internet protocol stack. However, they are capable enough to use a protocol stack specifically designed for them. In particular, they have enough resources to provide support for security functions and thus can be integrated as fully developed peers into the Internet.

Class 2 devices are less constrained and might be capable of employing a full Internet protocol stack. However, since these devices typically need

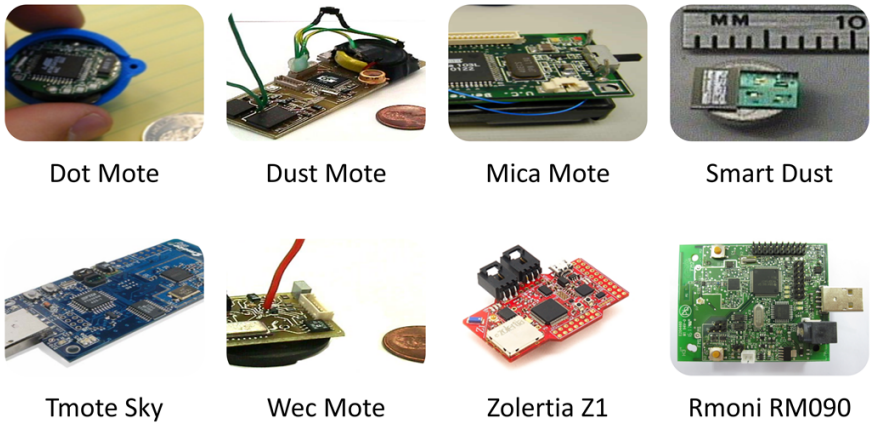


Figure 1.5: Sample sensor nodes (motes).

to run some kind of application, reducing the footprint of the Internet protocol stack leaves more resources available to the applications.

For the sake of comparison, at the time of writing this dissertation, a typical smart phone contains at least 1 GB of RAM and 8 GB of Flash. These values are about 100,000 times more than a class 1 device. Please note that Moores law [1] tends to be less effective for embedded devices than in personal computing devices. Any gains made possible by increasing the transistor density of the embedded devices are more likely to be invested in the further reduction of these devices' cost and power requirements than into increasing their computing resources.

Sensor nodes are also referred to as motes. Several motes that can be reprogrammed exist on the market. Figure 1.5 shows some examples of such motes. During this PhD research we have used the Tmote Sky [7] and the Zolertia Z1 [8] as representatives for class 1 devices, and the Rmoni RM090 [9] as a representative for class 2 devices.

1.1.1.3 IoT networks

Sensors and actuators do not operate in isolation, but are connected. They often form their own networks and communicate wirelessly in most cases. Such a network is called Wireless Sensor and Actuator Network (WSAN) or just Wireless Sensor Network (WSN) for the sake of simplicity. WSNs are an extension of the current Internet architecture as illustrated in Figure 1.6. WSN are typically connected to the Internet by Gateways (GWs). These gateways have to be able to

[1] "Moore's law" is the observation that, over the history of computing hardware, the number of transistors in a dense integrated circuit has doubled approximately every two years.

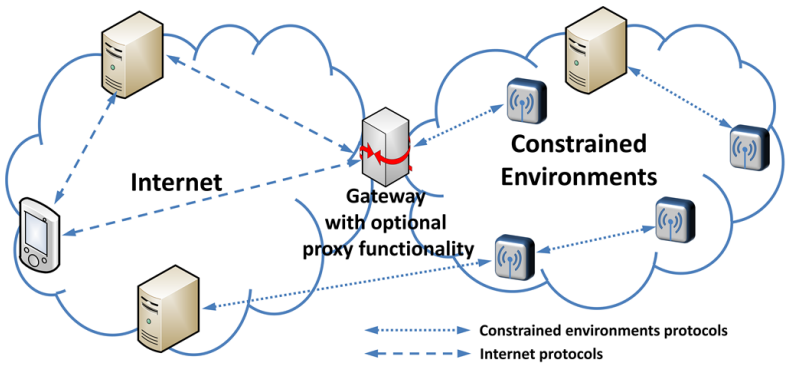


Figure 1.6: Constrained Environments architecture

communicate between the Internet protocol stack and the WSN protocol stack and to translate between them as needed.

Similar to the constrained devices that they connect, WSNs are also constrained and have different characteristics than those typical in today's Internet. These constrained networks have high packet loss, low throughput, frequent topology changes and small useful payload sizes. They are referred to as Low-power and Lossy Network (LLN).

1.1.2 Open standards stack for the IoT

The integration of embedded devices into the Internet introduces new challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. Therefore, it is in most cases unfeasible to run standard Internet protocols on this class of constrained devices and/or LLNs. New or adapted protocols that take into consideration the capabilities of the constrained devices and the characteristics of LLNs, are required.

In the past years, there were many efforts to enable the extension of the Internet technologies to constrained devices. In the absence of widely accepted standard protocols for constrained devices, many vendors developed – out of necessity – proprietary protocols to run inside their sensor networks. Connectivity between the Internet and the sensor networks was achieved through the use of vendor-specific gateways or proxies. These gateways have to translate between protocols used in the Internet and proprietary protocols used in the sensor networks. Figure 1.7 displays two different sensor networks that are connected to the Internet by gateways. Users on the Internet have to connect to the gateways in order to obtain data from the corresponding sensor network. There are several ways how a gateway can handle such user requests. For example, the gateway from vendor 1 translates standard Internet protocols into proprietary sensor protocols and relays the requests to the

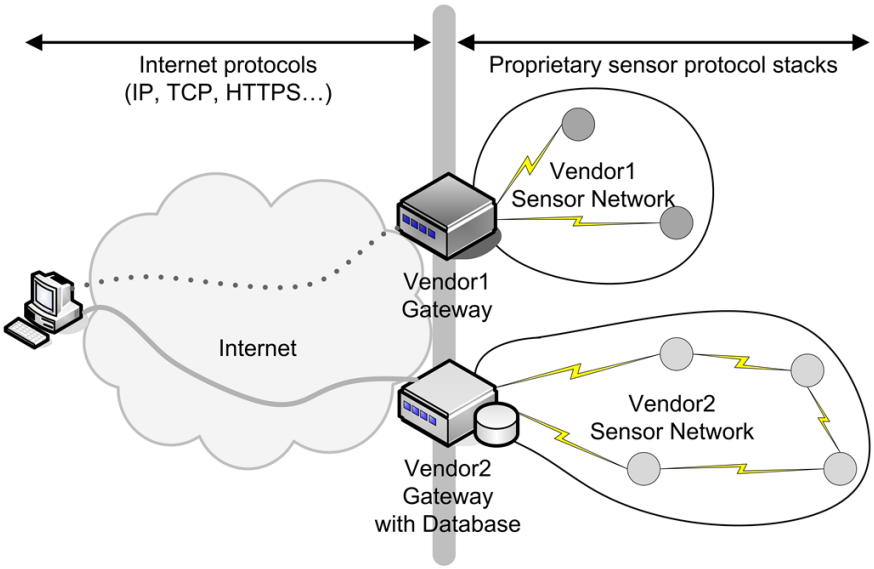


Figure 1.7: Gateways and proprietary protocols are often used to interconnect sensor networks to the Internet.

sensors in its network. The gateway then receives the answers from the relevant sensors by means of the proprietary sensor protocols and sends back the appropriate reply to the user using standard Internet protocols. The gateway offers an API that applications should use in order to create requests that can be understood by the gateway. This approach has the benefit that direct (real-time) interaction with sensor nodes is possible, but only by using a vendor-specific interface. Alternatively, the gateway of vendor 2 contains a database with pre-collected sensor data. When it gets a request from a user on the Internet, it replies directly to the requester using the data in the database. In some cases, the gateway is simply running a web server that makes the data available to the outside world. In this case, existing database technologies can be reused, but the user does not know whether the returned data is coming from the sensors in real-time or whether it is coming from information that has been previously stored in a database.

The use of standardized solutions is mostly limited to the use of a standard for the physical layer and MAC layer, although tailored MAC protocols could be used as well. It is clear that such an approach hinders the integration of sensors into the Internet. Little flexibility is offered since users can query the sensors only in the way that is allowed by the gateway. Another disadvantage is the vendor lock-in: gateways and sensors often have to be from the same vendor in order to be compatible. In addition, creating and maintaining the gateway requires significant development effort: often even adding new sensor resources requires making

(administrative) changes to the gateway. Finally, due to the lack of real end-to-end connectivity, no real-time interaction with the constrained devices is supported.

These limitations in combination with the general understanding that constrained devices will take up a prominent role in the future Internet, raised the need for standardized, open solutions for network communication with constrained devices. To ensure wide adoption, these new solutions have to be interoperable with the most widely used protocols in the Internet, initially Internet Protocol (IP) and, in a later stage, Hypertext Transfer Protocol (HTTP). To address these needs, the Internet Engineering Task Force (IETF) – responsible for the development of high-quality Internet standards – has formed several working groups. Initial IETF working groups focused on the networking layer: *IPv6 over Low Power WPAN* (6LoWPAN) [10] and *Routing Over Low Power and Lossy Networks* (ROLL) [11]. The 6LoWPAN group tackles the transmission of IPv6 packets over IEEE 802.15.4 networks and the ROLL group develops IPv6 routing solutions for LLNs.

However, the expansion of the Internet in the 90s was not due to introducing new or better networking protocols. It was a result of introducing the World Wide Web (WWW), which made it easy to integrate services and applications. One of the essential technologies underpinning the WWW was the HTTP protocol. Today, HTTP has become a key protocol in the realization of scalable web services building around the Representational State Transfer (REST) paradigm. The REST architectural style enables the realization of scalable and well-performing services using uniform and simple interfaces. The availability of an embedded counterpart of HTTP and the REST architecture could boost the uptake of the IoT.

Therefore, more recently, work started to allow integration of constrained devices in the Internet at service level. The IETF Constrained RESTful Environments (CoRE) working group has realized the REST architecture in a suitable form for the most constrained nodes and networks [12]. To that end the Constrained Application Protocol (CoAP) was introduced, a specialized RESTful web transfer protocol for use with constrained networks and nodes. CoAP realizes a subset of REST that is common with HTTP, but is optimized for Machine-to-Machine (M2M) applications.

Together, these protocols allow the IP-based integration of constrained devices into the Internet in a standardized way, as shown in Figure 1.8. Similar to the previous approaches, gateways are still used to translate between the protocols used in the Internet and protocols used in the sensor networks, e.g., IPv6 to 6LoWPAN and vice versa. However, due to the use of standardized protocols, many of the disadvantages from the previous approaches are now solved. For example it is now possible to combine sensor devices from different vendors in the same network, or to use a gateway from a different vendor than the vendor of the sensor devices. Flexibility is also improved by this approach as users are not confined to the API offered by the gateway: users can directly query the sensors without the need for

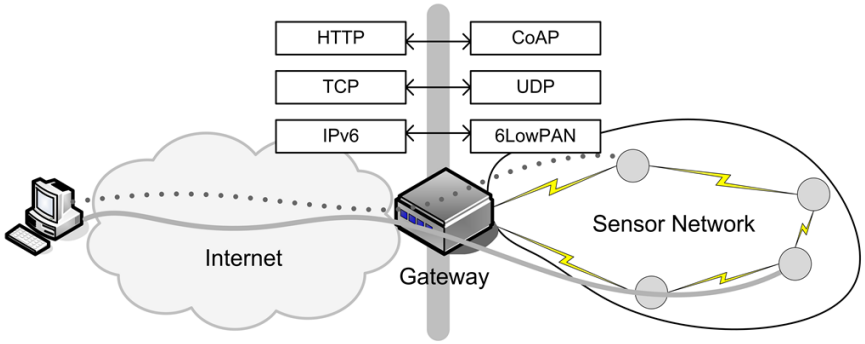


Figure 1.8: Internet protocols are extended to the sensor networks. The Gateway translates between the two standardized protocol stacks.

the gateway that understands the query or needs to interpret the data. The application payload can now travel directly from the client to the sensor, where it is processed and acted upon. The gateway takes care of the translation between standardized protocols. This end-to-end approach makes adding and removing sensor resources transparent to the gateway and improves interoperability of devices.

Combined, the individual standards form an *IETF LLN protocol stack* to support the realization of an interoperable Internet-of-things. In Figure 1.9 a representation is given of how the different LLN standards fit together. Due to the popularity of IEEE 802.15.4, this standard is often used at the physical layer (PHY) and the Media Access Control (MAC) layer.

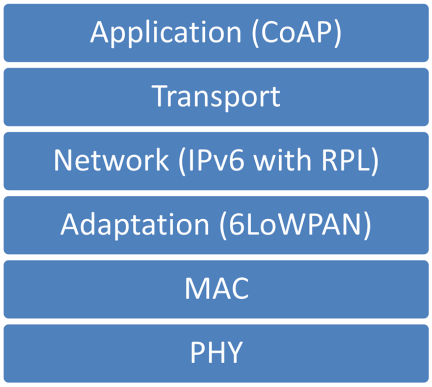


Figure 1.9: IETF LLN protocol stack

1.2 Challenges

This PhD research builds upon CoAP to enable a better integration of constrained devices in the IoT and examines proposed CoAP solutions theoretically and experimentally proposing alternatives when appropriate. The overall goal of this PhD research is to develop enablers on top of CoAP to facilitate the creation of applications. This dissertation addresses the following main IoT challenges to achieve this goal:

Challenge 1: Use of open standards. As described in Section 1.1.2, many vendors developed proprietary protocols to run inside their sensor networks because widely accepted standard protocols for constrained devices were missing. Vendor-specific gateways have to translate between protocols used in the Internet and the proprietary protocols used in the sensor networks. As a result, users have limited flexibility for expansion and development of their networks, since they are limited to the options offered by a particular vendor.

Challenge 2: Service discovery and self-organization. With recent technologies, it has now become possible to deploy a sensor network and interconnect it with IPv6 Internet. Within the sensor network itself, the available protocols are largely self-organizing, requiring no human intervention. Also, if the IPv6 address of a sensor is known, its resources can be accessed using CoAP. Nevertheless, there are several gaps related to the automatic discovery of sensors, integration with current Internet standards such as DNS, user-friendly access to sensors from within a web browser or the fact that several manual configuration steps are still needed to integrate a sensor network within an existing networking environment.

Challenge 3: Interactions with a multitude of objects and resources. Depending on the application, information from individual objects might not be sufficient, reliable, or useful. An application may need to aggregate and/or compare data from a group of objects in order to obtain accurate results. Likewise, it is often needed to control more than one actuator at once to make the complete object act as desired. Although multicast may be used to transmit the same request to several objects, multicast communication with smart objects has some disadvantages. Programming individual requests is another solution but lacks flexibility and opportunities for reusability.

Challenge 4: Local data processing. Sometimes, data from many sources needs to be collected and processed before it can be used. Often it is not desirable to transmit large amounts of data or continuous data

streams to far away Cloud services for reasons of latency, explosion of data traffic, reliability, *etc*. Therefore, current trends in research and industry try to bring processing closer to the data sources [13, 14].

Challenge 5: Deployment. WSN are often using the Industrial, Scientific and Medical (ISM) bands, which are used by many other devices. Thus the WSN devices have to coexist and compete with them to use the same medium. As the number of devices increases and their usage increases problems related to congestion become more relevant. A good understanding about the performance of today's open standards in realistic environments and at larger scale is lacking.

1.3 Outline

This dissertation is composed of a number of publications that were realized within the scope of this PhD. The selected publications provide an integral and consistent overview of the work performed. The different research contributions are detailed in Section 1.4 and the complete list of publications that resulted from this work is presented in Section 1.5.

Within this section we give an overview of the remainder of this dissertation and explain how the different chapters are linked together. Fig. 1.10 positions the different contributions that are presented in each chapter (Ch.) and appendix (App.).

We start by proposing a mechanism that facilitates the deployment of sensor networks and enables the discovery, end-to-end connectivity and service usage of newly deployed sensor nodes (Chapter 2). The proposed approach makes use of CoAP and combines it with Domain Name System (DNS) in order to enable the use of user-friendly Fully Qualified Domain Names (FQDNs) for addressing sensor nodes. It includes the automatic discovery of sensors and sensor gateways and the translation of HTTP to CoAP, thus making the sensor resources globally discoverable and accessible from any Internet-connected client using either IPv6 addresses or DNS names both via HTTP or CoAP. As such, the proposed approach provides a feasible and flexible solution to achieve hierarchical self-organization with a minimum of pre-configuration. By doing so we minimize costly human interventions and eliminate the need for introducing new protocols dedicated for the discovery and organization of resources. This reduces both cost and the implementation footprint on the constrained devices.

The larger part of this dissertation focuses on using CoAP to realize communication with groups of resources. In many IoT application domains, sensors or actuators need to be addressed as groups rather than individually, since individual resources might not be sufficient or useful. A simple example is that all lights in a

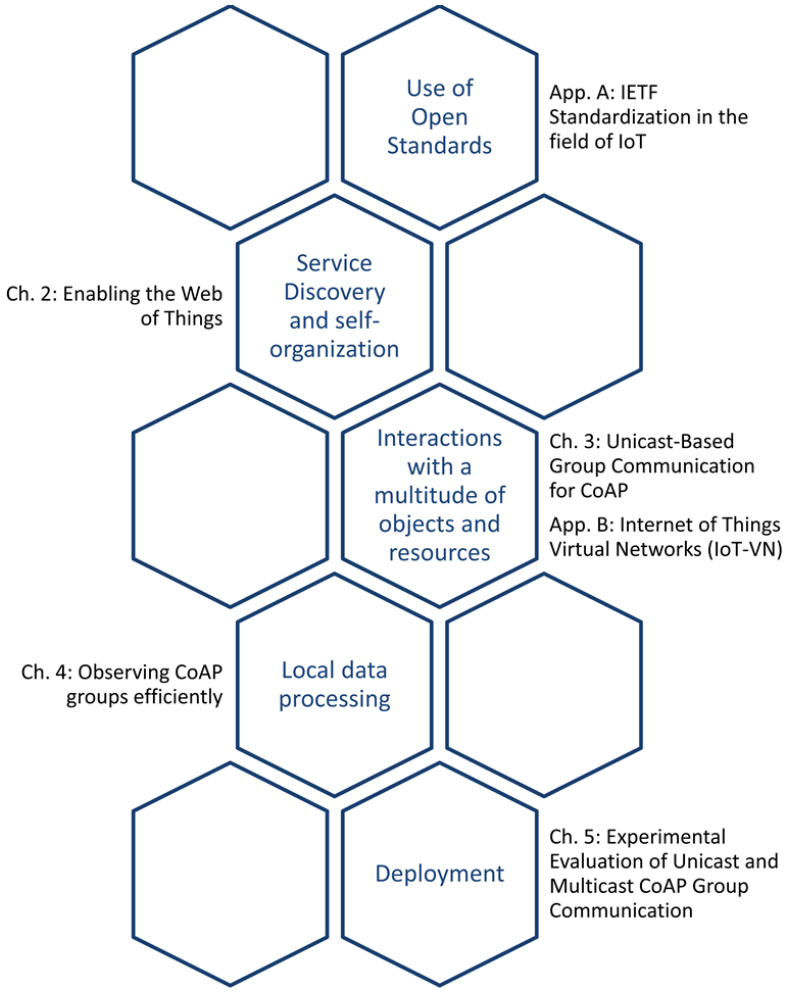


Figure 1.10: Schematic position of the different chapters in this dissertation

room should go on or off as a result of the user toggling the light switch. As not all IoT applications may need group communication, the CoRE working group did not include it in the base CoAP specification. This way the base protocol is kept as efficient and as simple as possible so it would run on even the most constrained devices. Group communication and other features that might not be needed by all devices are standardized in a set of optional separate extensions. We first examine the proposed CoAP extension for group communication, which utilizes Internet Protocol version 6 (IPv6) multicasts. We highlight its strengths and weaknesses and propose our own complementary solution that uses unicast to realize group

communication. Our solution offers capabilities beyond simple group communication. For example, we provide a validation mechanism that performs several checks on the group members, to make sure that combining them together is possible. We also allow the client to request that results of the individual members are processed before they are sent to the client. For example, the client can request to obtain only the maximum value of all individual members (Chapter 3).

Another important optional extension to CoAP allows clients to continuously observe resources by registering their interest in receiving notifications from CoAP servers once there are changes to the values of the observed resources. By using this publish/subscribe mechanism the client does not need to continuously poll the resource to find out whether it has changed its value. This typically leads to more efficient communication patterns that preserve valuable device and LLN resources. Unfortunately CoAP observe does not work together with the CoAP group communication extension, since the observe extension assumes unicast communication while the group communication extension only support multicast communication. In Chapter 4, we propose to extend our own group communication solution that was introduced in Chapter 3 with group observation capabilities. By combining group observation with group processing features, it becomes possible to notify the client only about certain changes to the observed group (e.g., the maximum value of all group members has changed).

Acknowledging that the use of multicast as well as unicast has strengths and weaknesses we propose to extend our unicast based solution with certain multicast features. By doing so we try to combine the strengths of both approaches to obtain a better overall group communication that is flexible and that can be tailored according to the use case needs (Chapter 5). Together, the proposed mechanisms represent a powerful and comprehensive solution to the challenging problem of group communication with constrained devices. We have evaluated the solutions proposed in this dissertation extensively and in a variety of forms. Where possible, we have derived theoretical models and have conducted numerous simulations to validate them. We have also experimentally evaluated those solutions and compared them with other proposed solutions using a small demo box and later on two large scale wireless sensor testbeds and under different test conditions. The first testbed is located in a large, shielded room, which allows testing under controlled environments. The second testbed is located inside an operational office building and thus allows testing under normal operation conditions. Those tests revealed performance issues and some other real problems. We have provided some solutions and suggestions for tackling those problems.

Apart from the main contributions, two other relevant outcomes of this PhD research are described in the appendices. In Appendix A, we review the most important IETF standardization efforts related to the IoT and show that with the introduction of CoAP a complete set of standard protocols has become available

to cover the complete networking stack and thus making the step from the IoT into the Web of Things (WoT). Using only standard protocols makes it possible to integrate devices from various vendors into one big WoT accessible to humans and machines alike.

In Appendix B, we provide an alternative solution for grouping constrained devices by using virtualization techniques. Our approach focuses on the objects, both resource-constrained and non-constrained, that need to cooperate by integrating them into a secured virtual network, named an Internet of Things Virtual Network or IoT-VN. Inside this IoT-VN full end-to-end communication can take place through the use of protocols that take the limitations of the most resource-constrained devices into account. We describe how this concept maps to several generic use cases and, as such, can constitute a valid alternative approach for supporting selected applications.

Table 1.2 shows the challenges that were highlighted in Section 1.2 and indicates which were targeted per chapter.

Table 1.2: An overview of the contributions per chapter in this dissertation.

	Chapter				Appendix	
	2	3	4	5	A	B
Use of open standards	•	•	•	•	•	
Service discovery and self-organization	•					
Interactions with a multitude of objects and resources		•	•	•		•
Local data processing		•	•	•		
Deployment	•			•		

1.4 Research contributions

In Section 1.2, the problems and challenges for using CoAP to integrate constrained devices into the Internet are formulated. They are tackled in the remainder of this dissertation for which the outline is given in Section 1.3. To conclude, we present an elaborated list of the research contributions within this dissertation:

- Design and implementation of a CoAP based resource discovery protocol.
  - Implementation on non-constrained devices using the CoAP++ framework [15]. The CoAP++ framework was realized using Click Router [16], a C++ based modular framework that can be used to realize any network packet processing functionality.

  - Implementation on constrained devices using the IDRA framework [17], a network architecture and application platform developed for TinyOS [18] and written in nesC.

- Design and implementation of an alternative CoAP group communication protocol.

  - Implementation of a unicast based group communication solution including creation, validation, behavior manipulation, observe support, and simple local data processing.
  - Implementation on non-constrained devices using Click Router.
  - Implementation of constrained devices using Contiki [19] and Erbium [20].

- Experimental test of multicast- and unicast-based group communication approaches under a variety of test conditions.

  - Large-scale (40 sensors) tests under controlled conditions using a testbed in a large shielded room [21].
  - Medium-scale (10 sensors) tests under typical office environment [22].
  - Small-scale (6 sensors) tests using a home automation demo box [23].
  - Simulations using COOJA the Contiki network simulator.

- Contribution to ongoing CoAP standardization efforts.

  - Coauthored the Internet Draft: ”CoAP Profile Description Format draft-greevenbosch-core-profile-description”.
  - First author of a new Internet Draft: ”CoAP Entities draft-ishaq-core-entities”.
  - Participated in CoAP plugtests to examine interoperability of various CoAP implementations [24, 25].

1.5 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during my PhD research.

1.5.1 Publications in international journals (listed in the Science Citation Index [2])

1. **Isam Ishaq**, Jeroen Hoebeke, Jen Rossey, Eli De Poorter, Ingrid Moerman, and Piet Demeester. *Enabling the web of things: facilitating deployment, discovery and resource access to IoT objects using embedded web services.* Published in the International Journal of Web and Grid Services, Volume 10, issue 2–3, pp. 218–243, 2014.

2. **Isam Ishaq**, Jeroen Hoebeke, Floris Van den Abeele, Jen Rossey, Ingrid Moerman, and Piet Demeester. *Flexible unicast-based group communication for CoAP-enabled devices.* Published in Sensors Magazine, Volume 14, issue 6, pp. 9833–9877, 2014.

3. **Isam Ishaq**, Jeroen Hoebeke, Ingrid Moerman, and Piet Demeester. *Observing CoAP groups efficiently.* Published in Ad Hoc Networks Journal, Sep. 2015.

4. **Isam Ishaq**, Jeroen Hoebeke, Ingrid Moerman, and Piet Demeester. *Experimental evaluation of unicast and multicast CoAP group communication.* Submitted to the Journal of Network and Computer Applications, Aug. 2015.

1.5.2 Publications in other international journals

1. **Isam Ishaq**, David Carels, Girum Ketema Teklemariam, Jeroen Hoebeke, Floris Van den Abeele, Eli De Poorter, Ingrid Moerman, and Piet Demeester. *IETF standardization in the field of the Internet of Things (IoT): a survey.* Published in the Journal of Sensor and Actuator Networks, Volume 2, issue 2, pp. 235–287, 2013.

1.5.3 Publications in international conferences (listed in the Science Citation Index [3])

1. **Isam Ishaq**, Jeroen Hoebeke, Ingrid Moerman, and Piet Demeester. *Internet of things virtual networks: bringing network virtualization to resource-*

[2] The publications listed are recognized as 'A1 publications', according to the following definition used by Ghent University: A1 publications are articles listed in the Science Citation Index Expanded, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper.

[3] The publications listed are recognized as 'P1 publications', according to the following definition used by Ghent University: P1 publications are proceedings listed in the Conference Proceedings Citation Index - Science or Conference Proceedings Citation Index - Social Science and Humanities of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper, except for publications that are classified as A1.

constrained devices. Published in proceedings of the 2012 IEEE International conference on green computing and communications, conference on Internet of things, and conference on cyber, physical and social computing (GreenCom 2012), 20–23 Nov. 2012, pages 293–300, Besançon, France, 2012.

2. **Isam Ishaq**, Jeroen Hoebeke, Floris Van den Abeele, Ingrid Moerman, and Piet Demeester. *Group Communication in Constrained Environments using CoAP-based Entities.* Published in proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2013), 21–23 May 2013, pages 345–350, Cambridge, Massachusetts, USA, 2013.

1.5.4 Publications in other international conferences

1. **Isam Ishaq**, Jeroen Hoebeke, Jen Rossey, Eli De Poorter, Ingrid Moerman, and Piet Demeester. *Facilitating sensor deployment, discovery and resource access using embedded web services.* Published in proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012), 4–6 July 2012, pages 717–724, Palermo, Italy, 2012.

2. Jeroen Hoebeke, David Carels, **Isam Ishaq**, Girum Ketema Teklemariam, Jen Rossey, Eli De Poorter, Ingrid Moerman, and Piet Demeester. *Leveraging upon standards to build the Internet of things.* Published in proceedings of the 19th IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT 2012), 16 Nov. 2012, pages 1–6, Eindhoven, Netherlands, 2012.

3. Floris Van den Abeele, Jeroen Hoebeke, **Isam Ishaq**, Girum Ketema Teklemariam, Jen Rossey, Ingrid Moerman, and Piet Demeester. *Building embedded applications via REST services for the Internet of Things.* Published in Proceedings of the 11th ACM Conference on Embedded Network Sensor Systems, 11–15 Nov. 2013, Roma, Italy, 2013.

1.5.5 Contributions to standardization bodies

1. **Isam Ishaq**, Jeroen Hoebeke, Floris Van den Abeele. *CoAP Entities draft-ishaq-core-entities-00.* Published as IETF Internet Draft, 17 Jun. 2013.

2. Bert Greevenbosch, Jeroen Hoebeke, **Isam Ishaq**, Floris Van den Abeele. *CoAP Profile Description Format draft-greevenbosch-core-profile-description-02.* Published as IETF Internet Draft, 21 Jun. 2013.

1.5.6 Patent applications

1. **Isam Ishaq**, Jeroen Hoebeke. *Entity creation for constrained devices.* WO Patent App. PCT/EP2013/078,055, Koninklijke Kpn N.V., Iminds, University Gent, Jul. 2014.

References

[1] *iMinds insights - The Internet of Things*, 2014. [Online; accessed 8 May 2015]. Available from: http://www.iminds.be/nl/inzicht-in-digitale-technologie/iminds-insights.

[2] D. Minoli. *Building the internet of things with IPv6 and MIPv6: The evolving world of M2M communications*. John Wiley & Sons, 2013.

[3] D. D. Guinard and V. M. Trifa. *Building the Web of Things*. Manning, in press.

[4] K. Ashton. *That internet of things thing*. RFiD Journal, 22(7):97–114, 2009.

[5] T. Ray. *Internet of Things: Mammoth Morgan Stanley Note Tries to Explain It*. Tech Trader Daily, 26, 2013.

[6] C. Bormann, M. Ersue, and A. Keranen. *Terminology for Constrained-Node Networks*. RFC 7228 (Informational), May 2014. Available from: http://www.ietf.org/rfc/rfc7228.txt.

[7] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. *Deploying a wireless sensor network on an active volcano*. Internet Computing, IEEE, 10(2):18–25, 2006.

[8] *Zolertia Z1 Platform*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.zolertia.com/ti.

[9] *Rm090 — RMONI wireless nv*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.rmoni.com/en/products/hardware/rm090.

[10] *IPv6 over Low power WPAN (6lowpan)*, 2012. [Online; accessed 28 December 2012]. Available from: http://datatracker.ietf.org/wg/6lowpan/.

[11] *Routing Over Low power and Lossy networks (roll)*, 2012. [Online; accessed 3 October 2012]. Available from: http://datatracker.ietf.org/wg/roll/.

[12] *Constrained RESTful Environments (core)*, 2014. [Online; accessed 19 March 2014]. Available from: http://datatracker.ietf.org/wg/core/.

[13] *Cisco Fog Computing with IOx*, 2015. [Online; accessed 5 March 2015]. Available from: http://www.cisco.com/web/solutions/trends/iot/cisco-fog-computing-with-iox.pdf.

[14] F. Van den Abeele, J. Hoebeke, G. K. Teklemariam, I. Moerman, and P. Demeester. *Sensor Function Virtualization to Support Distributed Intelligence in the Internet of Things*. Wireless Personal Communications, pages 1–22, 2015.

[15] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester. *Facilitating sensor deployment, discovery and resource access using embedded web services*. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, pages 717–724. IEEE, 2012.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. *The Click modular router*. ACM Transactions on Computer Systems (TOCS), 18(3):263–297, 2000.

[17] E. De Poorter, E. Troubleyn, I. Moerman, and P. Demeester. *IDRA: A flexible system architecture for next generation wireless sensor networks*. Wireless Networks, 17(6):1423–1440, 2011.

[18] *TinyOS*, 2012. [Online; accessed 3 April 2012]. Available from: http://www.tinyos.net/.

[19] *Contiki: The Open Source Operating System for the Internet of Things*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.contiki-os.org/.

[20] *Erbium (Er) REST Engine and CoAP Implementation for Contiki*, 2015. [Online; accessed 27 February 2015]. Available from: http://people.inf.ethz.ch/mkovatsc/erbium.php.

[21] *wilabt*, 2015. [Online; accessed 3 May 2015]. Available from: http://ilabt.iminds.be/wilabt.

[22] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester. *The w-iLab. t testbed*. In Testbeds and Research Infrastructures. Development of Networks and Communities, pages 145–154. Springer, 2011.

[23] F. Van den Abeele, J. Hoebeke, I. Ishaq, G. K. Teklemariam, J. Rossey, I. Moerman, and P. Demeester. *Building embedded applications via REST services for the Internet of Things*. In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, page 82. ACM, 2013.

[24] *1st CoAP Plugtest*. Technical Report, April 2012.

[25] L. Velez. *IoT COAP2 Interop Event Preliminary Report*, 2012. [Online; accessed 28 December 2012]. Available from: http://svn.tools.ietf.org/svn/wg/core/Preliminary-Results-CoAP2.pdf.

2

Enabling the Web of Things: Facilitating deployment, discovery and resource access to IoT objects using embedded web services

"People care about resources, nobody cares about discoveries these days."

– Robert Young

In this chapter, we propose a mechanism that facilitates the deployment of sensor networks and enables the discovery, end-to-end connectivity and service usage of newly deployed sensor nodes. The proposed approach makes use of CoAP and combines it with Domain Name System (DNS) in order to enable the use of user-friendly Fully Qualified Domain Names (FQDNs) for addressing sensor nodes. It includes the automatic discovery of sensors and sensor gateways and the translation of HTTP to CoAP, thus making the sensor resources globally discoverable and accessible from any Internet-connected client using either IPv6 addresses or DNS names both via HTTP or CoAP. At the time of doing this research in 2012, the standardization of the CoAP protocol was still in an early stage. Many optional extensions such as HTTP-CoAP proxying or discovery via a resource directory were not fully specified or implementations and their evaluation were not available. In order to discuss related works that have been published after the preparation of this article, this chapter includes an appendix that was not part of the published

journal version of the article.

⋆ ⋆ ⋆

I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester.

Published in the International Journal of Web and Grid Services, Vol. 10, Nos. 2/3, 2014.

Abstract As of now, the IETF Constrained Application Protocol (CoAP) is being standardized. CoAP takes the Internet of Things to the next level: it enables the implementation of RESTful web services on embedded devices, thus enabling the construction of an easily accessible Web of Things. However, before tiny objects can make themselves available through embedded web services, several manual configuration steps are still needed to integrate a sensor network within an existing networking environment. In this paper we describe a novel self-organization solution to facilitate the deployment of constrained networks and enable the discovery, end-to-end connectivity and service usage of these newly deployed sensor nodes. By using embedded web service technology, the need of other protocols on these resource constrained devices is avoided. It allows automatic hierarchical discovery of CoAP servers, resulting in a browsable hierarchy of CoAP servers, which can be accessed both over CoAP and HTTP.

2.1 Introduction

The ubiquitous Internet protocol technology is rapidly spreading to new domains where constrained embedded devices such as sensors and actuators play a prominent role. This expansion of the Internet is comparable in scale to the spread of the Internet in the '90s and the resulting Internet is now commonly referred to as the Internet of Things (IoT). The integration of embedded devices into the Internet introduces several new challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. These embedded devices are typically designed for low cost and power consumption and thus have very limited power, memory, and processing resources and are often disabled for long-times (sleep periods) to save energy. The networks formed by these embedded devices are also constrained and have different characteristics than those typical in today's Internet. These constrained networks have high packet loss, low throughput, frequent topology changes and small useful payload sizes.

In the past few years, there were many efforts to enable the extension of the Internet technologies to constrained devices. Most of these efforts were focus-

ing on the networking layer: IPv6 over Low-Power Wireless Personal Area Networks (RFC4919) [1], Transmission of IPv6 Packets over IEEE 802.15.4 Networks (RFC4944) [2], RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks Routing (RFC6550) [3] or the ZigBee adoption of IPv6 [4]. These standardization efforts enable the realization of an Internet of Things, where end-to-end connectivity with tiny objects such as sensors and actuators becomes possible.

However, the great success of the current Internet was not caused by solely supporting global connectivity. Only with the advent of web service technology, the development of all kinds of interesting services became possible and the World Wide Web became a reality. Today, (RESTful) web service technology is at the basis of many successful companies. A similar embedded counterpart of web service technology is needed in order to exploit all great opportunities offered by the Internet of Things and turn it into a Web of Things. Recently, standardization work has started to allow precisely the integration of constrained devices with the Internet at the service level. The Constrained RESTful Environments (CoRE) working group is in the process of realizing the Representational State Transfer (REST) architecture in a suitable form for the most constrained nodes and networks. To that end, the Constrained Application Protocol (CoAP) was introduced, a specialized RESTful web transfer protocol for use with constrained networks and nodes [5]. CoAP realizes a subset of REST that is common with the Hypertext Transfer Protocol (HTTP), but is optimized for Machine-to-Machine (M2M) applications.

With these technologies, it has now become possible to deploy a sensor network, to interconnect it with IPv6 Internet and to build applications that interact with these networks using embedded web service technology. Within the sensor network itself, the available protocols are largely self-organizing, requiring no human intervention. Also, if the IPv6 address of a sensor is known, its resources can be accessed using CoAP. Nevertheless, there are still several important hurdles that need to be overcome. Several gaps exist with regard to the automatic discovery of sensors, integration with current Internet standards such as DNS, user-friendly access to sensors from within a web browser or the fact that several manual configuration steps are still needed to integrate a sensor network within an existing networking environment. However, the advent of open standards for embedded web services on e.g. sensors and sensor gateways, offers new opportunities to tackle several of these challenges related to the deployment of sensor networks and the realization of global user-friendly connectivity and access to sensor resources by making use of embedded web services through the CoAP protocol.

In this paper, we will describe novel self-configuration and bootstrapping mechanisms in order to facilitate the deployment of sensor networks and enable the discovery, end-to-end connectivity and service usage of newly deployed sensor nodes. The proposed approach makes use of CoAP and combines it with DNS in order to enable the use of user-friendly FQDN for addressing sensor nodes. It includes the

automatic discovery of sensors and sensor gateways and the translation between HTTP and CoAP, thus making the sensor resources globally discoverable and accessible from any Internet-connected client using either IPv6 addresses or DNS names both via HTTP or CoAP. As such, the proposed approach provides a feasible and flexible solution to achieve hierarchical self-organization with a minimum of pre-configuration. It bridges the gap between the deployment of constrained objects and the actual consumption of their services by users, services or other machines.

Section 2.2 discusses how the on-going work in the IETF CoRE working group contributes to the realization of the Web of Things. Next, we give an overview of the challenges related to sensor network deployment, discovery and access (Section 2.3). In Section 2.4 we then introduce a solution based on CoAP and DNS for the hierarchical self-configuration of sensors and access via HTTP, followed by a brief discussion in Section 2.5 about all new possibilities that are unleashed when tiny objects can be easily deployed, integrated with the Internet and made accessible via embedded web service technology. Section 2.6 presents actual deployment results and in Section 2.7 the performance of the deployment is analyzed. Section 2.8 gives an overview of related work. The paper is concluded in Section 2.9.

2.2 From IoT to WoT

Recent research on embedded web services is laying the groundwork for a better integration of sensor resources into the service web and, as such, provides the foundations for realizing what is called the Web of Things (WoT). Since the dominating web protocol HTTP is too complex, the IETF CoRE working group, formed in 2010, has designed a simpler web protocol - CoAP. CoAP uses the same RESTful principles as HTTP, but it is much lighter so that it can be run on constrained devices [6, 7]. As a result, CoAP has a much lower header overhead and parsing complexity than HTTP. It uses a 4-bytes base binary header that may be followed by compact binary options and a payload. Optional reliability is supported within CoAP itself by using a simple stop-and-wait reliability mechanism upon request. Secure communication is also supported through the optional use of Datagram Transport Layer Security (DTLS).

The CoAP interaction model is similar to the client/server model of HTTP. A client can send a CoAP request, requesting an action specified by a method code (GET, PUT, POST or DELETE) on a resource (identified by a URI) on a server. The CoAP server processes the request and sends back a response containing a response code and payload. Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport such as UDP and thus it also supports multicast CoAP requests. This allows CoAP to be used for point-to-multipoint interactions which are commonly required in automation.

At the time of writing this article, the CoAP protocol was not finalized yet. However it is considered in its final stages before being finalized. Nevertheless several implementations of the CoAP protocol for various platforms and programing languages already exist; many of which run Class 1 and Class 2 devices as classified in [8]. Class 1 devices are resource-constrained devices with ˜10KiB of RAM, and ˜100KiB of ROM. Class 2 devices have ˜50KiB of RAM, and ˜250KiB of ROM. Interoperability between many of these implementations has been formally tested by the European Telecommunications Standards Institute (ETSI) at two events called ETSI IoT CoAP Plugtests. In addition to assessing the interoperability of participating products, these Plugtests events aimed to validate the CoRE base standards. The existence of such a wide range of implementations across a broad range of programing languages and most importantly platforms demonstrates the feasibility of the protocol implementation. For an overview of the CoAP base protocol and related drafts we refer to [9].

CoAP and HTTP have been compared several times in the literature. For example, [10] compared CoAP to HTTP in terms of mote's energy consumption by simulations for a fixed 10 second client request interval. The results show that while receiving and processing packets, the energy consumed when using CoAP is approximately half compared to the one consumed when using HTTP. While transmitting packets, the energy required by CoAP was 4 times lower than the energy required by HTTP. When testing the response time on real sensor motes, this study shows that using CoAP over UDP introduces 9 to 10 fold lower response times than HTTP over TCP. Similarly, [11] reports that CoAP/UDP perform better than HTTP/TCP for the intelligent cargo container use case that was evaluated. In particular the author reports a 6 times lower message size and a 4 times lower Round Trip Time (RTT). This is mainly due to CoAPs compressed header and the avoidance of the TCP handshaking mechanisms. The study of [12] shows that generally UDP based protocols perform better for constrained networks due to using lower number of messages when retrieving resources. But even when both protocols (CoAP and HTTP) are run over UDP, the same study shows that CoAP performs better than HTTP in constrained environments. CoAP also has the added value of providing optional reliability since it has its own simple retransmission capability.

The IETF CoRE working group considers constrained RESTful environments as an extension of the current web architecture as illustrated in Figure 2.1. The group envisions that CoAP will complement HTTP and that CoAP will be used not only between constrained devices and between servers and devices in the constrained environment, but also between servers and devices across the Internet [13]. An important requirement of the CoRE working group is to ensure a simple mapping between HTTP and CoAP so that the protocols can be proxied transparently. This proxy functionality is often implemented on the gateways that interconnect

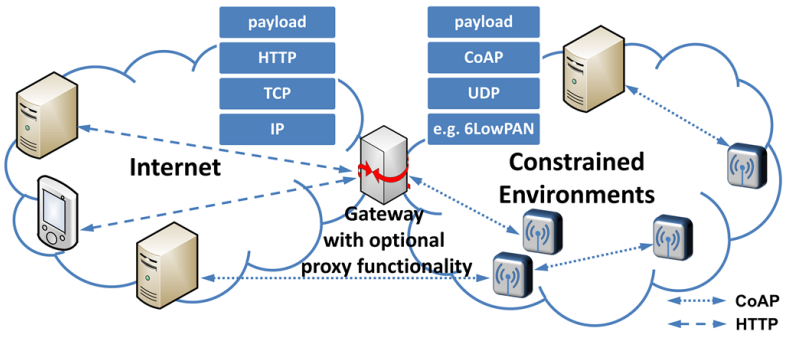


Figure 2.1: Constrained Environments architecture and the CoAP protocol stack.

the constrained environments to the Internet. Thus gateways play a central role in the constrained environments architecture as can be seen in Figure 2.1. These gateways have to be able to communicate between the Internet protocol stack and the constrained environments protocol stack (see Figure 2.1) and to translate between them as needed. In the remainder of this article we will use the term *Gateway (GW)* for the device that is located at border of constrained network and providing routing and communication functionality. We will use the term *proxy* whenever we talk about translation between protocols or whenever data is being cached.

Resource discovery is important for M2M interactions, and is supported in CoAP using the CoRE Link Format [14]. A well-known URI "/.well-known/core" is defined as a default entry-point for requesting the list of links about resources hosted by a CoAP server. Once the list of available resources is obtained from the server, the client can send further requests to obtain the value of a certain resource. The example in Figure 2.2 shows a client requesting the list of the available resources on a server (GET /.well-known/core). The returned list shows that the server has, amongst others, a resource called /s/t that would return back the temperature in degrees Celsius. The client then requests the value of this resource (GET /s/t) and gets a reply back from the server (23.5C).

2.3 Problem statement

The deployment of sensor networks, including their integration in the Internet, is a multi-faceted problem. First of all, there is the deployment of the sensor network itself, starting with the provisioning of the hardware, followed by the actual installation and optimal placement of the sensing infrastructure [15] and potentially, calibration. Once installed and activated, it is up to the communication protocols to create a fully operational sensor network that is robust, energy efficient and capable of communicating to and from the sensor gateway.

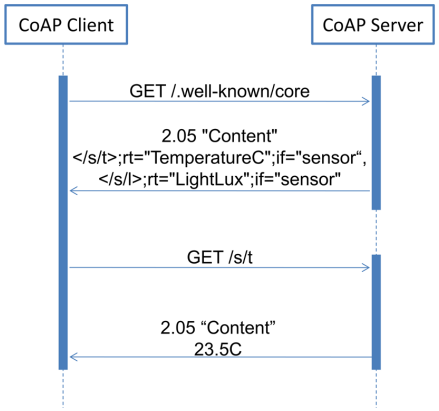


Figure 2.2: An example of resource discovery and access using CoAP.

Looking at the literature and standardization bodies (see Section 2.1), it is clear that there have been many efforts to create such protocols, including MAC layer protocols, addressing, routing, data collection protocols, *etc*. [16]. As such, sensor network self-configuration, i.e. the creation of an operational sensor network and communication inside the sensor network can be considered as a well-studied problem for which several solutions exist.

The next aspect is connectivity with the IP-based Internet. On the one hand, sensor networks using proprietary networking solutions can be integrated with the Internet by deploying appropriate gateways with proxy functionality to be able to translate from and to the sensor network protocols. On the other hand, there is significant momentum for IP-based sensors and actuators as illustrated by the IETF work mentioned in Section 2.1 and in several research papers. As such, the feasibility of integrating sensor networks with the Internet and enabling IP-based connectivity to sensors has been shown and made possible, for example in [17–20].

However, this is only the starting point. Next to the connectivity within the sensor network and the connectivity with the Internet, there are many other aspects related to the deployment of sensor networks. When a sensor subnet is connected to the Internet it needs to receive an address prefix, routing to the sensor network should be configured; ideally it should integrate with current Internet standards such as DNS, *etc*. Typically, manual interventions are still needed by an administrator. In addition, connectivity can be achieved, but knowing which sensors are present, discovering them and being able to use them in a user-friendly way that does not require any technical skills (e.g. from a web browser) is an interesting challenge that has only begun to receive more attention from the research community recently. It is clear that there are still many open aspects and challenges. In this paper, we describe a novel solution that is capable of dealing with several

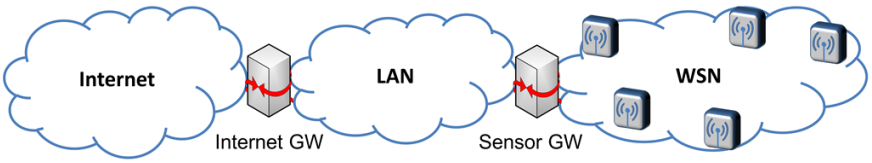


Figure 2.3: Sample network topology used to present the self-organization solution. A WSN is connected via a sensor gateway to a LAN, which in turn is connected to the Internet via an Internet gateway.

of these challenges. To this end, we have taken a fresh approach, making use of embedded web services.

2.4 Solution

By making use of the functionalities offered by CoAP, we have designed a hierarchical self-configuration solution that facilitates the deployment, discovery and resource access for sensor networks. In this section, we will present our approach in more detail.

2.4.1 Assumptions

In order to be able to design a self-configuration solution, one always has to make a few assumptions about certain aspects that have been preconfigured already. For example, in order for a new PC to auto-configure its globally routable IPv6 address, a router advertisement daemon has to be active in the network announcing the prefix of the network. Of course, the challenge is to restrict the required amount of pre-configuration involving humans as much as possible and to avoid these configuration steps at deployment time of the devices that dynamically join the network.

To present our self-configuration solution, we assume a basic network topology as shown in Figure 2.3 (more complex topologies are possible, but are out of the scope of this paper). An organization is connected via an Internet gateway, which also acts as DNS server, to the IPv6 Internet and has obtained a /48 IPv6 range and suffix for its domain names (e.g. “test.ibbt.be”). From this /48 range, a network administrator can assign subnets to different networks. For example, a /64 subnet is assigned to the LAN network behind the Internet GW (e.g. “iot.test.ibbt.be”). Now assume the organization wants to equip its building(s) with wireless sensors, which will be connected to this LAN network via one or more sensor gateways. The administrator reserves a pool of /64 subnets, domain name suffixes and sensor gateway names that can be assigned to newly deployed sensor networks.

We consider the sensors as dumb devices that only have a minimal knowledge. For our self-configuration solution, we make the following assumptions. The sensor knows or will discover its address in the sensor network. Typically, because of the limitations of a sensor device, these addresses are preferably small, e.g. only 16-bit for 6LoWPAN short addresses [21]. The complete IPv6 address of the sensor is not known, since it also depends on the sensor network the node will be deployed in. In the remainder of the paper, we will assume the use of unique, manually configured 16-bit addresses for the sensors. The assignment or generation of these unique addresses is out of the scope of this paper. Further, we assume the sensor knows its (or a) name. This name is a variable-length string and could be anything, for example a hardware identifier. A user-friendly name such as temperature_room1 would require user intervention and knowledge about the location where the sensor will be deployed. This can be done after deployment, where the automatically generated name can be replaced by a more meaningful name. Finally, the sensor runs a minimal CoAP server. Since the proposed solution makes use of CoAP, this is a strict requirement. Further, this minimal CoAP server should offer a well-known resource (/.well-known/serverInfo) that allows the retrieval of its name and address. No assumptions are made about the protocols inside the sensor network. These can be standardized or proprietary and it is the responsibility of the sensor gateway to translate from the IPv6 world to the sensor world. For the sensor gateway we also assume it will run a CoAP client and server and that it knows its global IPv6 address in the LAN.

2.4.2 The solution

Based on these assumptions, we have designed a complete self-organization solution which is summarized in Figure 2.4. Please note that for simplicity, resolving of DNS names is ignored in this figure. Also note that the CoAP servers, indicated between [], can use either DNS names or IPv6 addresses. As can be seen in this figure, our solution is organized in three phases: self-configuration, sensor discovery and sensor access (by a client or by a resource directory server).

After the sensors have been deployed, the sensor gateway discovers the sensors in its network by sending a multicast (ff02::1) CoAP GET request for the resource /.well-known/serverInfo. The response sent by this resource is currently a plain text string that contains whatever information the sensor knows about itself according to the following proprietary format:

AL|*addressLength*|A|*address*|NT|*nameType*|N|*name*|SID|*subnetID*|SP|*prefix*|SM|*subnet*|SD|*domain*

Each sensor replaces the italic texts in the above format with its own information. As a minimum the sensor would send its address length (AL), its address (A), its name type (NT, S for short names and L for long names), and its name. The

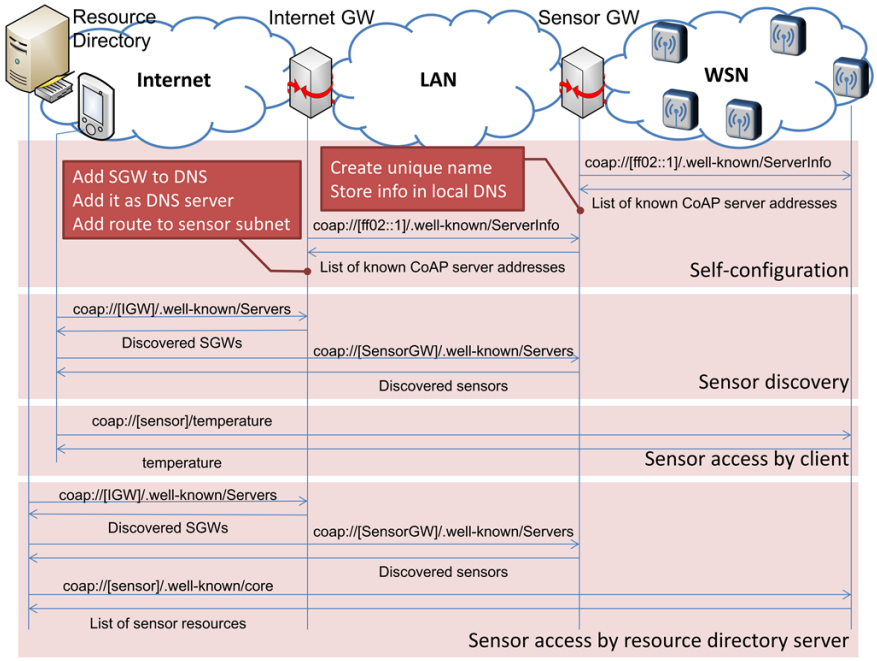


Figure 2.4: Complete self-organization process, sensor discovery and resource access by a client and by a resource directory server.

other information describes the subnet in which the sensor is located and might or might not be known by the sensor itself. Ideally, in the future, a standardized format should be used, possibly using a structured representation such as JSON.

When receiving the replies from the sensors to the multicast request, the sensor gateway is able to discover the (short) address and name of all sensor nodes present in the network. To verify whether already discovered sensors have disappeared (or new ones have been deployed), the sensor gateway can be configured to periodically perform the multicast. Of course, in this case the multicast frequency should be limited in order to limit the resulting energy consumption. Since discovery is triggered by the gateway, we refer to this approach as a pull-based solution. Section 2.7 will describe an alternative method we have implemented where sensor nodes announce their presence to the gateway, once or periodically (push versus pull-based solution).

Assuming the sensor gateway has obtained a subnet prefix and domain for the sensor network (see next paragraph), it can construct the complete IPv6 address and FQDN of each sensor. This information is then used to dynamically update the local DNS running at the sensor gateway (note that the sensor gateway acts as resolver of DNS requests for names in the sensor domain). When a sensor is

no longer available i.e. it does not reply to a certain number (3 by default) of consecutive periodic discovery broadcasts, its information is removed from the local DNS. It is important to note that these updates to the DNS are restricted to the sensor gateway and stay within the administrative domain of the company.

The same discovery process can be repeated at a higher level in the network hierarchy assuming that the sensor gateways also run CoAP servers. The Internet gateway can periodically send CoAP multicast requests for /.well-known/serverInfo in the LAN network in order to discover all sensor gateways. The resource /.well-known/serverInfo of a sensor gateway will also contain, in addition to the address and name of the sensor gateway, the domain suffix of the sensor subnet and the IPv6 prefix of the sensor subnet. In a similar way, the Internet gateway adds the address and name of the discovered sensor gateways to its local DNS. In addition, the Internet gateway dynamically installs a route to the sensor subnet and adds the sensor gateway as the name server for the sensor network. In case the Internet gateway notices that the sensor gateway does not have a subnet prefix, domain suffix and name configured, the Internet gateway will take this information from its pool (see our assumptions) and send it as a CoAP POST request to the sensor gateway, which will update its configuration accordingly. Please note that the order of discovery, first sensors and then sensor gateways, can be the other way around.

This discovery process can be repeated for different levels in the networking hierarchy up to the highest level, which, in our simple example, is the Internet gateway. Now, everyone in the Internet can resolve the FQDN of every discovered sensor and forward packets to this sensor. This means that all sensors are now globally reachable with minimal effort and end-to-end communication is now possible. Using the same principles, one can introduce additional levels of indirection in order to enhance scalability or realize more complex setups. At this point, CoAP can be used to e.g. update the name of the sensor or retrieve any other information such as measurements.

When a client wants to discover available sensors and make use of the services offered by sensor nodes, it now only has to know one anchor point for the entire domain of the organization (in a similar way a domain has a well-known name server). In our simple example, this is the Internet gateway, which could be assigned an easy to remember name such as coap.iot.test.ibbt.be. From that point on, a client can simply take the following actions:

- Send a CoAP request for the resource /.well-known/servers on the Internet gateway to get a list of all sensor gateways.
- Per sensor gateway, the client can send a CoAP request to /.well-known/servers in order to find all sensors in the attached sensor subnet.
- Per sensor, the client can now use CoAP to retrieve sensor information.

By applying this mechanism and creating a hierarchy of linked CoAP servers, any client (a human, another machine or a resource directory server) can easily discover and use any sensor without a lot of network overhead. For example, Figure 2.4 also shows how a resource directory can benefit from our solution in order to create a directory listing containing all embedded resources that are present in the network. In combination with the automatic creation of FQDN names for sensors and their addition to a DNS system, this creates a flexible discovery mechanism and enables user-friendly access to sensors for humans. The whole process is almost fully automated, minimizing human intervention.

2.4.3 HTTP access to sensors

The solution as described thus far enables access to sensors using DNS names and IPv6 addresses using CoAP. However many clients do not have a CoAP implementation and will therefore not be able to benefit from this solution. On the other hand all web client implementations have a web browser that supports HTTP. Since CoAP is following the same RESTful principles as HTTP, both protocols can be nicely mapped to each other [22] and thus making the sensor resources accessible via HTTP. To achieve this mapping, HTTP-to-CoAP proxy functionality is required. In addition, to enable real browsable discovery of and access to sensor resources, we have foreseen a translation mechanism to create HTML pages from responses in the CoRE Link Format. Both mechanisms are explained in this subsection.

To enable HTTP access in our solution, the sensor gateway and the Internet gateway were extended in such a way to not only act as CoAP servers, but also as HTTP-to-CoAP proxies capable of translating HTTP messages to CoAP messages and vice versa. Clients can access these gateways via their favorite web browser using HTTP requests. The gateways map the requests to CoAP and send the requests to the sensors. Once the sensor replies using CoAP, the reply is sent back to the client using HTTP and the client remains unaware of the fact that CoAP was used to retrieve the reply from the sensor.

The implemented proxy application on the gateways can act in two modes: transparent and non-transparent. In the non-transparent mode the client should construct the HTTP request in the following format:
http://gw_name:8080/sensor_name/resource.

The gateway then translates this request into the following CoAP request and sends it to the respective sensor: coap://sensor_name/resource. It is clear that in this non-transparent mode, the client must explicitly be aware of the proxy and use it as part of the URI. In the future, we also want to foresee non-transparent proxying based on the CoAP Proxy-Uri option as described in [14]. Using this method, the CoAP (or HTTP) request is sent to the proxy and the CoAP Proxy-Uri

option gives the absolute URI of the actual resource to be queried.

In the transparent mode the client remains unaware of the presence of the proxy functionality on the gateway and constructs the HTTP request in the following format: http://sensor_name:8080/resource. For this proxy to work properly in transparent mode, the proxy has to be on the path between the client and the sensor in order to be able to intercept the HTTP request (and TCP connection) and map it into the appropriate CoAP request. In our example, the transparent proxy functionality for accessing the sensors resides only on the sensor gateways. When the HTTP request for the sensor enters, the proxy will behave as the end point of the TCP connection and will handle the TCP connection. In the background, a translation to CoAP takes place and the request is sent to the sensor. For the user it seems as if he connects directly to the sensor using HTTP/TCP, but in reality the sensor gateway transparently handles the connection and translates it to CoAP. As such, in transparent mode, the user does not have to be aware of a proxy that it needs to use.

In addition to the mapping between HTTP and CoAP, the proxy implementation on the gateway performs automatic rewriting of response in the CoRE Link format into HyperText Markup Language (HTML). This extension does not provide any benefit for M2M communication but is aimed towards humans. By rewriting the CoRE Link format into HTML the information can be interpreted directly by the web browser and easily understood by humans. Every resource in a response in the CoRE Link Format, such as </sensors/temp> is rewritten by the proxy into an HTML link. When the original request made use of a proxy, the HTTP URI will consist of the proxy address or name, followed by the address or name of the actual CoAP server on which the resource resides and the resource itself. When the original request did not make use of a proxy or transparent proxying is possible, the HTTP URI will only contain the actual CoAP server on which the resource is located followed by the resource. For a more detailed description of our implementation we refer to [23].

2.5 Next steps: basics for realizing Web of Things

Using the above solution for self-configuration, newly deployed constrained devices are automatically discovered, their names are added to DNS and their resources are directly accessible and browsable over IPv6 via HTTP or CoAP or can be added to a directory server. This solution therefore presents an important building block that facilitates the actual usage of embedded web services as is required for building the Web of Things. Once end-to-end access to embedded web services has been realized, adding new functionalities or building novel services involving IoT objects is straightforward.

For example, the state of resources can be continuously observed using the

CoAP observe extension [24], leading to an as consistent as possible representation of resources. Using conditional observations [25], interested parties can be notified about resource states that satisfy specific conditions, thereby acting as an enabler to build applications such as sensor – actuator interactions. These extensions enrich the capabilities of the basis CoAP protocol and contribute to the realization of the Web of Things.

Looking at the resources themselves, several representations can be explored, ranging from plain text formats over formats defined by the Internet Protocol for Smart Objects (IPSO) alliance [26] to complex semantic representations using ontologies that are adapted to the specific applications and domains as described for instance in [27]. Also, the SPITFIRE project is providing vocabularies to describe sensors and to integrate them with the Linked Open Data cloud. Semantic descriptions of embedded web services allow linking sensor data with other available data. It brings the potential of semantic web technology (e.g. searching and reasoning) to constrained devices, realizing a Semantic Web of Things.

Further, through embedded web services, existing web service technologies, tools and frameworks become reusable for building web application and web services that are based on the state of the real world. However, it will impose novel challenges to web service aspects that are currently well understood, such as e.g. web service composition, due to the limitations of the constrained devices. For instance, embedded web services can be composed to create complex interaction scenarios where knowledge about the real world is used, linked with other services and processed to act again upon the physical world. Existing composition and orchestration frameworks such as described in [28], need to be extended in order to allow the incorporation of embedded web service technology and to realize the WoT. Also, when time varying data from constrained objects is incorporated or web services act upon the real world, issues such as consistency, failures, correct execution of all transactions as described in [29] need to be explored in view of a constrained environment.

Using standardized technologies, powerful and scalable solutions for collecting, storing and processing a multitude of sensor data can be developed. The link with state-of-the-art Cloud technology solutions that are gradually being adopted is clear [30]. Also tiny objects can be introduced as part of grid computing e.g. for the collection and processing of environmental information. In [31] an extensive overview of the introduction of mobile devices into Grid systems is given and an extension to the constrained world seems feasible with the advent of embedded web service technology.

Similar to search engines in the World Wide Web, sensor resources could be indexed just as regular web pages and made available to Internet users. Of course, issues such as time dependent aspects should be taken into consideration (e.g. indexing a temperature sensor) introducing novel challenges and opportunities.

This short discussion clearly reveals the great potential offered through the availability of embedded web service technology. It can really facilitate the realization of WoT services, opening up access to sensor data and stimulating their widespread usage, while at the same time avoiding vertically integrated and closed systems. As such, it presents great opportunities to researchers active as well in the field of web service technology as in the field of embedded distributed systems.

2.6 Deployment

Together, the mechanisms described in Section 2.4, realize a hierarchical self-configuration solution based on CoAP. Automatically discovered CoAP servers, up to the sensor level, are linked together into a browsable hierarchy that can be accessed either via CoAP or HTTP, offering global access to sensor resources in a human-friendly way through the use of names. The described solutions have been implemented and deployed on a publicly reachable testbed that includes both real and simulated sensor nodes. The implementation consists of two parts, the implementation running on the sensor nodes and the implementation running on the gateways.

The implementation on the gateways, called CoAP++, has been largely realized in Click Router, a C++ based modular framework that can be used to realize any network packet processing functionality [32]. It consists of several modules such as the CoAP protocol, a CoAP server backend capable of offering resources, CoAP server discovery with DNS integration, HTTP-to-CoAP proxying, USB sensor communication, *etc*. These modules can be combined in several ways by creating a configuration file. As such, using the same code base, one can realize the following configurations:

- A stand-alone socket-based CoAP client making use of IPv6/UDP sockets for network communication.
- A stand-alone packet-based CoAP client that processes and generates complete network packets (including Ethernet/IPv6/UDP headers) offering full control over the communication which is interesting for the realization of the gateway functionality.
- A CoAP sensor gateway with sensor discovery and DNS integration, and (non-) transparent proxy functionality. The sensor discovery module in the gateway discovers the sensors in the sensor network and, based on the subnet and domain info it has obtained statically or from the Internet gateway, it creates the FQDN name and the mapping between IPv6 addresses and names. For example if the sensor node with short address 1 and name node1 is discovered by the sensor gateway and the sensor gateway is responsible

for the sensor subnet aaaa::/64 and sensor domain subnet1.iot.test.ibbt.be, the FQDN name node1.subnet1.iot.test.ibbt.be is created and mapped to the IPv6 address aaaa::1. Next, this module interacts with dnsmasq, which runs locally on the gateway. First, as soon as the subnet and domain info are known, dnsmasq is configured in such a way that the sensor gateway acts as the forward and reverse DNS server for the sensor subdomain and subnet. Next, when sensors are discovered and the FQDN-IPv6 mapping has been created, the service discovery module dynamically updates the dnsmasq configuration file, adding the forward and reverse DNS records, and signals dnsmasq about any changes. This way, it is made sure that the DNS server in the sensor gateway always has up-to-date information.

- A CoAP Internet gateway with sensor gateway discovery and DNS integration and non-transparent proxy functionality. When the sensor gateway discovery module discovers sensor gateways the following will happen. If discovered sensor gateway already knows its subnet and domain, this information, together with the IPv6 address of the gateway, is used to update the dnsmasq configuration on the Internet gateway. The sensor gateway is designated as the forward and reverse DNS server for the sensor subnet and all related DNS requests are forwarded to this gateway. Note that the Internet gateway acts as the DNS server for all sensor subnets (which have a common domain). If the discovered sensor gateway does not know its subnet and domain, this information is dynamically assigned from a pool of domains and subnets. The information is sent back to the sensor gateway and dnsmasq on the Internet gateway is updated as described before. Finally, in both cases, also a new routing entry is installed. As such, all incoming IPv6 packets can be routed to the correct sensor subnet.

The CoAP client/server protocol and the CoAP resources on the sensors have been implemented using the IDRA framework [33]. IDRA is a network architecture and application platform developed for TinyOS[1] and written in nesC[2]. The designed solution for the sensors has a footprint of 37092 bytes in ROM and 5923 bytes in RAM. This footprint includes, in addition to CoAP, a simple (always on) Media Access Control (MAC) Protocol and the Ad hoc On-Demand Distance Vector (AODV) routing protocol [34]. Such a small footprint can be run on the most-constrained devices.

During our testing we had debug info enabled; consuming part of the 48kB ROM we had available. As such, we could only use the most basis MAC protocol, which is an always-on MAC. Thus we did not do an evaluation with a duty cycled MAC protocol. In the future, it should be possible to further evaluate our solution

[1] http://www.tinyos.net/
[2] http://nescc.sourceforge.net/

in a real duty cycled network when removing debug info or using new devices with e.g. 92kB ROM. Using another MAC will impact the performance of the solution. For example, when using an always-on MAC, the sender of a broadcast packet, only has to send the packet once, since every neighboring node should be able receive it immediately. When using a duty-cycled MAC such as Low Power Listening (LPL), the sender has to send the packet repeatedly during the entire duty cycle, consuming more energy for transmission and increasing the chance of collisions. On the other hand, less energy will be consumed because the radio will be sleeping most of the time. Next to this, there are several other MACs such as synchronized MAC protocols. Also, specialized MAC protocols for energy-efficient broadcasting in constrained networks exist as well [35]. The performance impact of different MACs on the presented solution is outside the scope of this work, but is interesting to explore in the future.

Access to the testbed and its resources is possible to anyone that is connected to the Internet using IPv6. The testbed can be accessed either over CoAP or over HTTP using the following URIs:

- coap://coap.iot.test.ibbt.be:5683/.well-known/core
- http://coap.iot.test.ibbt.be:8080/.well-known/core

To illustrate the browsable hierarchy of the solution a series of screenshots of HTTP access to the testbed is shown in Figure 2.5. For simplicity the screenshots skip the level of the Internet gateway and start at the level of a sensor gateway. With no prior knowledge about names, addresses or availability of the sensors and by using a standard web browser (without CoAP support) the client accessed the sensor gateway. In part (a) of Figure 2.5 the client was offered a list of available resources on the sensor gateway amongst which the client selected the “servers” resource to get a list of the known sensors running CoAP (b). The client selected one sensor “mote36” and got back a list of the available resource on that sensor (c). The client then selected the “m/t” resource to get back the measured temperature on that sensor in a semantic description as can be seen in part (d). In the entire process the client was communicating using HTTP and HTML, while the gateway was translating to and from CoAP and Core Link Format to be able to relay the communication to the sensors.

A sample of the direct end-to-end CoAP access to a resource on a sensor in the testbed is shown in Figure 2.6. The web browser used in this example was firefox[3] with the Copper (Cu) add-on[4] installed in order to be able to handle CoAP. In this case the sensor sent back the reply in CoRE Link Format as can be seen on the bottom right part of the figure. The add-on interpreted the CoRE Link format and displayed it visually on the left side of the figure.

[3] http://www.mozilla.org/firefox
[4] http://people.inf.ethz.ch/˜mkovatsc/copper.php

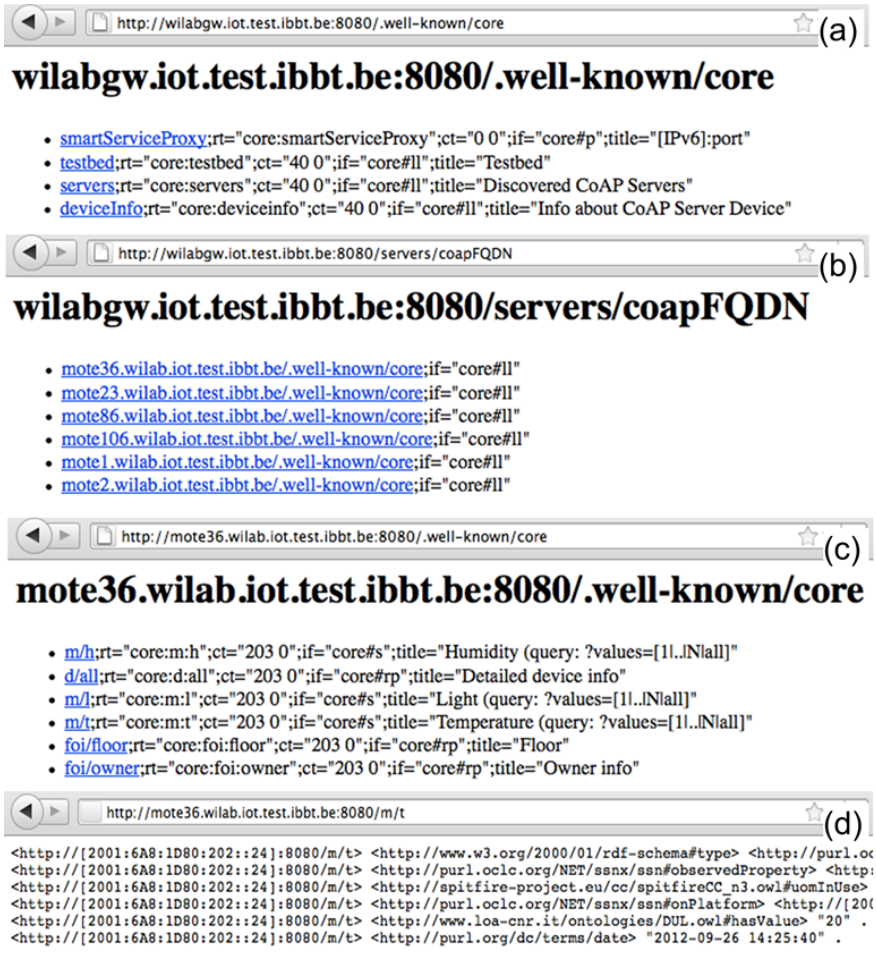


Figure 2.5: A step-by-step example illustrating the browsable hierarchy as a client using a web browser might follow to discover and access sensor resources.
(a) List of resource available on a Sensor Gateway.
(b) List of discovered Sensors running CoAP.
(c) List of resources offered by a particular sensor.
(d) The measured temperature on that sensor along its semantic description.

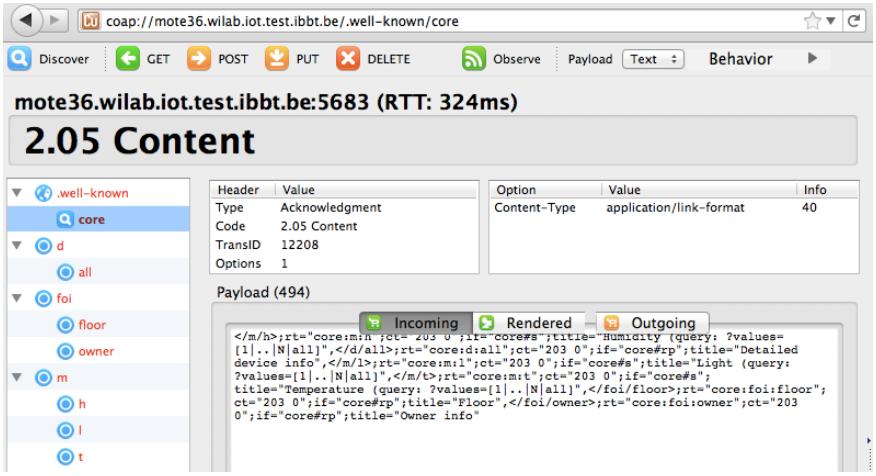


Figure 2.6: A sample of a direct end-to-end access to a CoAP resource on a sensor in the testbed using a web browser with the copper add-on installed. On the bottom right textbox the payload of the answer in CoRE Link Format is shown. On the left side a visual interpretation of this format is generated by the add-on.

In the above examples human friendly FQDN were used. As explained in Section 2.4, these names were automatically derived from the short sensor names and where automatically added to the relevant name servers. This is shown in Figure 2.7, where the discovered sensor information consisting of the name "mote36" and 16-bit address "36" has been used to create an IPv6 address and a FQDN have been dynamically added to the DNS on the sensor gateway. When a client queried the Internet gateway for the IPv6 for that sensor, the Internet Gateway forwarded the query to the appropriate Sensor Gateway, got a response from it and relayed the response back to the client.

2.7 Analysis of the different approaches

As mentioned in Section 2.4.2 we have implemented different approaches for the discovery of sensors inside the sensor network using CoAP. The sensor gateway can search for sensors (pull approach) or the sensors can announce their presence to the gateway (push approach). To validate these approaches in sensor discovery we conducted several experiments using a real-life wireless sensor network testbed, namely w-iLab.t [36]. This testbed is deployed across an operational office building with a significant number of co-located company wireless devices (Wi-Fi routers, DECT phones, Bluetooth devices, *etc.*) that cause interference to the sensors in the testbed and make the test networks lossy as is the case in realistic environments. The experiments were run on two sensor networks inside the

```
sh-3.2$ dig -t AAAA mote36.wilab.iot.test.ibbt.be

; <<>> DiG 9.8.1-P1 <<>> -t AAAA mote36.wilab.iot.test.ibbt.be
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17983
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;mote36.wilab.iot.test.ibbt.be. IN      AAAA

;; ANSWER SECTION:
mote36.wilab.iot.test.ibbt.be. 0 IN     AAAA    2001:6a8:1d80:202::24

;; AUTHORITY SECTION:
iot.test.ibbt.be.       339     IN      NS      dns.iot.test.ibbt.be.
```

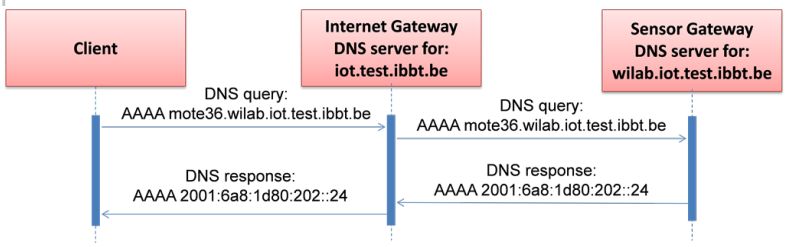


Figure 2.7: Name resolution using dynamically configured DNS information. The client queries for a sensor address (mote36.wilab.iot.test.ibbt.be). The request is received by the Internet gateway, which is the DNS server for iot.test.ibbt.be and forwards it to the Sensor Gateway since it is the DNS server for the subdomain wilab.iot.test.ibbt.be. The response consisting of the dynamically created IPv6 address from the short sensor address (36) is sent back to the client.

testbed with different network characteristics. The first sensor test network had 24 sensor nodes in a topology with a maximum broadcast domain of 22 nodes and nodes were within a maximum of 2 hops from the sensor gateway. The second test sensor network had 52 sensor nodes in a topology with a maximum broadcast domain of 38 nodes and nodes were within a maximum of 2 hops from the sensor gateway. In the following subsections we present the results of these experiments.

2.7.1 Pull approach

The advantage of the pull approach in discovering the sensors is that it can be manually executed whenever needed e.g. when the network is built or whenever it is modified. Manually triggered discovery can be complemented with periodic pulls to discover changes in the network and possibly generate alerts to the administrator. However the pull approach relies on sending multicasts, which are often translated into broadcasts inside sensor networks. Depending on the size and the topology of the network and on how broadcasts are forwarded into the network,

frequent broadcasts can easily lead to network congestion. Thus periodic pulls should be kept to a minimum and at relatively large intervals in order to avoid network congestion. Due to these potential congestions and to the fact that CoAP uses non-confirmable messages to reply to broadcasts, it is anticipated that some sensors might not get discovered immediately after the first discovery broadcast.

As anticipated, our tests revealed that using broadcasts as a means to discover the available sensors is challenging. In fact already with a broadcast domain of 22 nodes as was the case in our smaller test network, no sensors were initially discovered. The reason for that is that the sensors crashed as a result of sending just one discovery broadcast by the sensor gateway. This single broadcast packet was received by most sensors in the network and triggered those sensors to respond to it almost at the same time. Since the route to the destination of the response (the sensor gateway) was unknown to most sensors, the routing protocol on these sensors (AODV in our case) also started broadcasting to find routes to the sensor gateway. This ultimately led to a broadcast storm in the network and a receive buffer overflow in most sensors in the test network.

However in order to combat broadcast storms in the network and to improve the discovery rate, three techniques were used and examined. The first technique was to let the sensor gateway broadcast a route reply packet in the network. This caused all sensors to establish a route to the gateway and eliminated the need for them to start asking for a route once they get the discovery request. This way measuring the overhead of calculating routes to the gateway is avoided, since the evaluation of the used routing protocol is beyond the scope of this work. The other two techniques introduced random delays in the network – before responding the CoAP discovery requests and before forwarding broadcasts in the network.

On the top left corner of Figure 2.8 the effect of introducing random delays before sensors respond to discovery broadcasts on the percentage of discovered sensors for the two test networks is shown. In CoAP terms, this delay is called Leisure. The server could either use a default value for Leisure or compute a value for it. If the server has a group size estimate G, a target data transfer rate R and an estimated response size S, a rough lower bound for Leisure can then be computed as

$$Leisure_{lowerbound} = \frac{S \times G}{R} \tag{2.1}$$

For our two test networks G equals 22 and 38, S equals 100 bytes, and the target rate can be set to a conservative 8 kbit/s = 1 kB/s. The resulting lower bound for the Leisure is then equal to 2.2 and 3.8 seconds respectively. For a more complete discussion of the Leisure period and its estimation we refer to Section 8.2 of [5].

The graph shows that the average percentage of discovered nodes decreases with an increase of broadcast domains. The averages were computed for sending at least ten broadcasts for each experiment setting. The graph also shows that

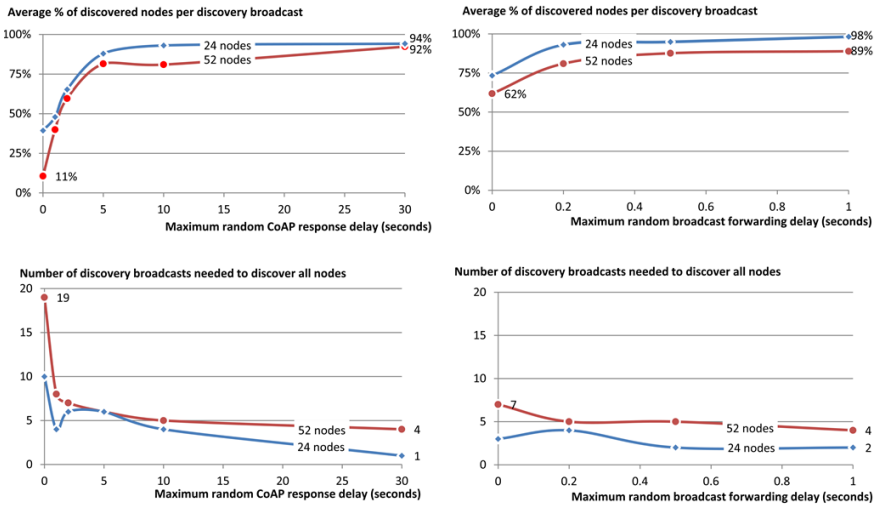


Figure 2.8: The effect of introducing random delays before sensors respond to CoAP discovery broadcasts and before they forward broadcasts to other sensors for two different network sizes. By randomly delaying the responds for a few seconds and the broadcast forwarding a few hundred milliseconds the discovery rate is highly improved.

already by randomly delaying the responses for a few seconds the discovery rate was highly improved. However it also shows that sending only one broadcast discovery CoAP request does not guarantee the discovery of all sensors in the network. This is due to the lossy nature of sensor networks and to the fact that broadcasts are not acknowledged and thus some sensors might not receive it. In the bottom left corner of Figure 2.8 the number of needed broadcasts to discover all sensors in the network is shown for the above experiments. The graph shows that by the introduction of a random delay of a few seconds all sensors are discovered significantly faster. Please note that these experiments were taken while having a maximum random rebroadcast delay of 200 milliseconds (see next paragraph).

On the right side of Figure 2.8 the effect of introducing a random delay before sensors forward any received broadcast to the other sensors on the discovery rate and on the number of needed requests to discover all sensors is shown. The graphs show that by delaying the rebroadcasting for several hundred milliseconds, the discovery rate is highly improved and thus fewer requests are needed to discover all sensors. Please note that these measurements were taken while having a Leisure of 10 seconds as explained in the previous paragraph.

Table 2.1 summarizes the results of the various experiments of the discovery following the pull approach and additionally shows the average discovery delays per sensor. Such delays are in most use cases for sensor discovery irrelevant and can be tolerated. Please note that all these measurements were taken after all

sensors have established a route as result of the broadcast route reply mentioned earlier in this section.

Table 2.1: Summary of the experiments showing the effect of changing the maximum random delays in forwarding broadcasts and in responding to CoAP discovery broadcasts (Leisure) on the discovery rate and delay for two different test networks. The averages were computed for sending at least ten broadcasts for each experiment setting.

Broadcast forwarding delay [sec]	Leisure [sec]	Network size [# of nodes]	Average discovery delay [sec]	Average discovered nodes per request [%]	# of BC to discover all nodes
0	0	24	Sensors crash due to broadcast storms		
		52			
	10	24	5.80	73%	3
		52	5.58	62%	7
0.2	0	24	1.69	39%	10
		52	1.08	11%	19
	1	24	1.87	48%	4
		52	3.00	40%	8
	2	24	2.31	65%	6
		52	3.00	60%	7
	5	24	3.31	88%	6
		52	4.46	81%	6
	10	24	5.72	93%	5
		52	5.62	81%	4
	30	24	13.98	94%	1
		52	12.68	92%	4
0.5	10	24	5.93	95%	2
		52	5.92	88%	5
1	10	24	5.91	86%	2
		52	5.45	89%	4

2.7.2 Push approach

As an alternative to the pull approach described in the previous subsection, it is possible to configure the sensors to announce their presence to the sensor gateway (push approach). To realize this, the sensors are hardcoded with an anycast address to which they send their announcement messages. This anycast address identifies a group of potential receivers. An anycast packet is routed in a similar way as a unicast packet until it reaches a receiver of the group. This first receiver will process the packet and stop its' forwarding. By configuring the gateway to listen to this anycast address, the gateway is able to receive all announcement messages. This approach has the advantage that the address of the sensor gateway does not have to be known by the sensors.

As mentioned in section 2.2, reliability of CoAP messages is optional. The sender can elect to either use confirmable or non-confirmable messages. By using confirmable messages, reliable delivery of the messages is guaranteed. Using non-confirmable messages reduces network traffic by eliminating the need for transmitting acknowledgments but does not guarantee that the messages are really delivered.

Our tests revealed that when using non-confirmable CoAP anycast messages the sensor discovery rate and thus the number of transmissions needed to make sure that the gateway has indeed discovered all sensors was very similar to using the pull approach. This is not surprising, since in both cases delivery of the messages depends on the load on the sensors and the network. When varying the random start-up delay of the nodes from 5 milliseconds to 5 seconds, we observed discovery percentages increasing from 79% up to 100%. A very low start-up delay means that most nodes will simultaneously announce their presence to the gateway, resulting in collisions. In this case, confirmable messages can help to alleviate this problem. However, it is expected that in real life situations, nodes will not start up almost simultaneously.

This push-based sending of anycast messages can be done once at boot time or periodically. Obviously when done only at boot time, there is a risk that the sensor gateway was not running or missed the registration and thus the sensor remains undiscovered. Periodically repeating the push-based sending of anycast messages is possible, but the frequency should be limited in order not to consume too much energy for sending and forwarding these messages. Of course, there will be a straightforward trade-off between energy consumption and speed of discovery. However this approach still makes sense to use when only a few sensors have been added to the network.

Another issue that should be considered when implementing the push approach on constrained devices is that the implementation will be slightly more complex and potentially have a higher footprint since it involves the use of timers in the code.

2.7.3 Pull-Push approach

The two discovery approaches can also be combined together as needed. For example one can use the pull approach for initial discovery of the network and for periodical re-discovery at relatively large time intervals. The administrator can also trigger a pull-based re-discovery when she thinks there are changes in the network. At the same time push-based notifications can be used so that sensors can announce their presence in the network, without waiting for the next pull cycle.

2.8 Comparison with related work

2.8.1 Related work

In this section we will discuss related work focusing on the automatic discovery of sensors, realization of end-to-end access and integration with DNS. Our solution is a network-based solution, meaning that we want to achieve global end-to-end access to sensor resources in a way that requires minimal to no human intervention. At the same time, we want to comply with current Web Standards by offering access using DNS and HTTP and we want to foresee means for the automatic discovery of sensors within a domain, since global access alone is not enough. Users should be able to find out which sensors are available.

Some solutions focus on the discovery of sensors and sensor data by publishing or collecting information about the sensors and measurements in databases that can be accessed by other applications and services. For example, Cosm[5] allows sensor data to be pushed to a central database, where it can be used by others to create applications. No direct access to sensors is allowed. The OGC SWE framework [37] defines web service interfaces for accessing sensor data, controlling sensors and alerting, functionalities comparable to the ones offered by CoAP. Reference [38] presents the OSIRIS Sensor Web Discovery Framework, which makes use of registries that are being built and which are capable of handling the dynamic properties of sensors. Similarly, a web crawler could periodically scan the Web of Things for sensors and downloading meta-data via their RESTful interfaces. This information can be stored, e.g. as Resource Description Framework (RDF) triples, after which the information can be searched, reasoned upon or linked with other open data [39].

These solutions aim for the realization of a Semantic Web of Things or Sensor Web, which enables data producers and users to publish and access sensor information via web- and standards based interfaces (see [40] for more details on Sensor Web). This goes already one step beyond our solution, since it focuses on the service part, skipping the deployment steps and ignoring the networking part. For example, aspects such as naming, automatic routing to sensor subnets, facilitating the deployment of sensor networks... are not considered, although very important. Our solution provides an answer to these problems, can be seen as a building block for the realization of the Semantic Web of Things and is therefore quite different, but complementary.

When focusing now more on the networking aspect (discovery followed by integration in DNS and automatic routing to sensor subnets), almost no related work is found. In [41], a solution is presented where a lightweight Multicast DNS - Service Discovery (mDNS-SD) implementation is running on a sensor platform. As

[5] http://www.cosm.com

such, sensor nodes can announce their services and update resource records in a DNS. However, this solution does not include other aspects such as the discovery and self-organization at higher hierarchical levels or the automatic configuration of routing to the subnet. Further, if RESTful web services want to be offered on top of the discovery, both an mDNS-SD and CoAP implementation are needed, increasing the code footprint. Taking a RESTful approach to tackle both problems mitigates this problem. In [42], a solution is presented for the integration of sensors and actuators in the Future Internet in a plug and play manner. Sensor nodes can register with gateways that provide an open interface to access raw or abstracted sensor data. At a higher level, a Sensor Address Server maintains a list of all registered gateways. This solution provides hierarchical levels, but does not comply with existing Internet standards nor foresees direct sensor resource access using IPv6. Finally, in [43], a zero-configuration IPv6/6LoWPAN-based system architecture is described. It foresees an API to access services following REST principles. A central unit can make use of this API to auto-discover the functionality offered by the sensor node or the service can be advertised in a way similar to multicast Domain Name System (mDNS). This approach uses embedded web services (not CoAP), but does not achieve the level of auto-configuration from our solution, i.e. multiple self-organizing hierarchical levels automatically linked with each other, resulting in a browsable discovery system that allows the discovery of and access to sensor resources. Also DNS aspects or automatic routing to sensor subnets, crucial in the actual roll-out of the network, are not considered.

2.8.2 High-level comparison

As already mentioned, the deployment of sensor networks, including their integration in the Internet, is a multi-faceted problem consisting of hardware provisioning, actual hardware installation and placement, optional calibration, the creation of an operational sensor network, *connectivity with existing networks, discovery, (user-friendly) access*, management and maintenance of the operational network. The solution proposed in this paper aims to facilitate and automate the steps marked in italic. For example, automatic DNS integration of sensors and sensors gateways and automatic routing towards the sensor network saves the installer time otherwise spent on manual configurations. After the discovery, an installer can easily interact with the discovered sensors in a RESTful way for further configuration, benefitting from features such as the DNS integration, HTTP/CoAP proxying and HTML translation.

As described in Section 2.8.1, there exist other solutions that fulfill part of the functionality required during deployment and the lifecycle of a sensor network. When using other solutions, an installer will still have to go through all the different steps inherent to any deployment scenario. Compared to solutions we are

aware of, our solution offers improvements in terms of automation: e.g. automatic DNS integration, automatic routing towards the sensor network. In addition it offers improvements in terms of footprint. Only a CoAP implementation is needed and no different protocols are being used for both discovery and resource access. As such, our solution nicely aligns with on-going standardization efforts for constrained networks in IETF and its footprint fits on the most constrained devices. This is different from e.g. solutions based on HTTP (which has too much overhead for a constrained network), mDNS (which requires additional protocols for resource access) or solutions coming from closed standardization bodies such as the Zigbee alliance.

2.9 Conclusions

In this paper we have described a novel self-organization solution to facilitate the deployment of sensor networks and enable the discovery, end-to-end connectivity and service usage of newly deployed sensor nodes. The proposed approach makes use of embedded web service technology, i.e. the IETF CoAP protocol. By combining it with DNS and foreseeing HTTP-to-CoAP proxy functionality, it complies with current Internet standards. Automatic hierarchical discovery of CoAP servers is one of the key features, resulting in a browsable hierarchy of CoAP servers, up to the level of the sensor resources. By creating a hierarchy of linked CoAP servers, scalability can be addressed. At this point, existing web crawler solutions can be used to index and publish the sensor information to the outside world. As such, the proposed approach provides a feasible and flexible solution to achieve hierarchical self-organization with a minimum of pre-configuration. The solution is based on a minimal number of assumptions regarding the pre-configuration. With some additional improvements and the development of management tools, it provides a valuable contribution to facilitate the deployment of and access to sensor networks. As such, it represents an important building block facilitating the actual usage of embedded web services as is required for building the Web of Things. Once end-to-end access has been realized, adding new functionalities or building novel services involving IoT objects is straightforward and many opportunities to integrate this with existing web service technologies arise.

The fact that embedded web services are used is a strong point, since it will facilitate integration with other services and applications. By complementing the solution with the appropriate firewalling and access policies, any level of sensor access can be made possible. The implementation and proper functioning of the solution has been demonstrated through the deployment on a publicly accessible test setup. This evaluation gave us insights about the strengths and limitations of the proposed approach in a large-scale, real-life environment. In the future, the solution will be further refined and tested, extended with other CoRE functionality

such as caching and issues such as the security of the presented solution will be investigated e.g. through the use of DTLS. In addition, it will be investigated to what extend any manual configuration that is still required can be avoided and how management tools based on embedded web service technology can bring the self-organization, configuration and management of sensor networks to a next level.

Acknowledgment

The research leading to these results has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 258885 (SPITFIRE project), from the iMinds ICON project OCareCloudS, from a VLIR PhD grant to Isam Ishaq and through an FWO postdoc research grant for Eli De Poorter.

Appendix

At the time of doing this research in 2012, the standardization of the CoAP protocol was still in an early stage. Many optional extensions such as HTTP-CoAP proxying or discovery via a resource directory were not fully specified or implementations and their evaluation were not available. In this appendix, we briefly discuss related works that have been published after the preparation of this article. This appendix was not part of the published journal version of the article.

The main IETF CoRE working group approach for resource discovery in constrained environments is described in the CoRE Resource Directory draft [44]. Although this document was still an Internet draft at the time of writing this appendix, this draft was actively discussed within the CoRE working group and has undergone several revisions. Therefore, we expect it will not undergo any major changes before becoming an RFC. The proposed approach relies on employing an entity called a Resource Directory (RD), which hosts descriptions of resources held on other servers. This allows others to perform lookups for those resources even when they are not available, e.g., when they are sleeping. Furthermore, this draft specifies web interfaces on the RD that servers can use to discover the RD and to register, maintain, lookup and remove resources descriptions. Additionally, this draft specifies a mapping between CoRE Link Format attributes and DNS Service Discovery (DNS-SD) fields. This allows RD entries to be easily exported to DNS-SD permitting discovery of CoAP services using DNS. In order to populate the RD, either a 3rd party has to discover all sensors and register them in the RD or the sensors themselves have to discover the RD (using multicast, anycast, via DHCP, etc.) upon which they can register their resources. As such the RD concept can be seen as complementary to our discovery mechanism and both approaches

can be integrated into an overall solution.

Jara et al. looked in [45] at the feasibility of using mDNS and DNS-SD in constrained networks. They argue that mDNS and DNS-SD cannot be directly applied in such networks and suggest a set of implementation guidelines and design recommendations for adapting them to become suitable for use in constrained networks. They called this approach Light-weight mDNS and DNS-SD (lmDNS-SD). One main recommendation is to reduce message sizes by avoiding the additional and authority records and by reducing the number of TXT records and use wild cards to compress them. Our work explored the integration of the sensor nodes into the DNS system, limiting the integration to the registration of name-IPv6 mappings (i.e. dynamic creation of AAAA records). We did not consider the integration of service discovery using DNS.

An alternative approach that also uses DNS-SD in proposed in [46]. However, instead of using mDNS, the authors developed their own discovery protocol EADP. EADP uses a trickle-inspired algorithm, which advertises a service only when an inconsistency appears. Intermediate nodes cache advertised services and can respond to queries on behalf of other nodes. Additionally, EADP optimizes, compresses, and enhances advertised DNS-SD messages to reduce their size. In contrast to mDNS, which uses multicast replies, EADP uses unicast replies. The authors report that EADP provides high discovery rates and fast discovery times with low resource consumption.

For an extensive and recent survey of discovery protocols for constrained M2M communications, we refer to [47].

References

[1] N. Kushalnagar, G. Montenegro, and C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. RFC 4919 (Informational), August 2007. Available from: http://www.ietf.org/rfc/rfc4919.txt.

[2] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC 4944 (Proposed Standard), September 2007. Updated by RFCs 6282, 6775. Available from: http://www.ietf.org/rfc/rfc4944.txt.

[3] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550 (Proposed Standard), March 2012. Available from: http://www.ietf.org/rfc/rfc6550.txt.

[4] *ZigBee IP.*, 2013. [Online; accessed 24 April 2013]. Available from: http://www.zigbee.org/Specifications/ZigBeeIP/Overview.aspx.

[5] Z. Shelby, K. Hartke, and C. Bormann. *Constrained Application Protocol (CoAP)*, 2012. [Online; accessed 3 November 2012]. Available from: http://tools.ietf.org/html/draft-ietf-core-coap-12.

[6] W. Colitti, K. Steenhaut, and N. De Caro. *Integrating wireless sensor networks with the web*. Extending the Internet to Low power and Lossy Networks (IP+ SN 2011), 2011.

[7] A. Dunkels et al. *Efficient application integration in IP-based sensor networks*. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pages 43–48. ACM, 2009.

[8] C. Bormann. *Guidance for Light-Weight Implementations of the Internet Protocol Suite*, 2012. [Online; accessed 3 November 2012]. Available from: http://tools.ietf.org/html/draft-ietf-lwig-guidance-02.

[9] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. V. d. Abeele, E. D. Poorter, I. Moerman, and P. Demeester. *IETF standardization in the field of the internet of things (IoT): a survey*. Journal of Sensor and Actuator Networks, 2(2):235–287, 2013.

[10] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota. *Evaluation of constrained application protocol for wireless sensor networks*. In Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on, pages 1–6. IEEE, 2011.

[11] T. Pötsch. *Performance of the Constrained Application Protocol for Wireless Sensor Networks*, 2011.

[12] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker, and C. Görg. *Implementation of coap and its application in transport logistics*. Proc. IP+ SN, Chicago, IL, USA, 2011.

[13] Z. Shelby. *Embedded web services*. IEEE Wireless Communications, 17(6):52, 2010.

[14] Z. Shelby. *Constrained RESTful Environments (CoRE) Link Format*. RFC 6690 (Proposed Standard), August 2012. Available from: http://www.ietf.org/rfc/rfc6690.txt.

[15] *White paper: Smart Networked Objects and Internet of Things*, 2012. [Online; accessed 3 November 2012]. Available from: http://www.instituts-carnot.eu/files/AiCarnot-White_Paper-Smart_Networked_Objects_and_Internet_of_Things.pdf.

[16] J. Zheng and A. Jamalipour. *Wireless sensor networks: a networking perspective*. John Wiley & Sons, 2009.

[17] M. Chen, S. Mao, Y. Xiao, and M. Li. *IPSA: a novel architecture design for integrating IP and sensor networks*. International Journal of Sensor Networks, 5(1):48–57, 2009.

[18] S. Duquennoy, N. Wirström, N. Tsiftes, and A. Dunkels. *Leveraging IP for Sensor Network Deployment*. In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+ SN 2011), Chicago, IL, USA, volume 11. Citeseer, 2011.

[19] J. W. Hui and D. E. Culler. *IP is dead, long live IP for wireless sensor networks*. In Proceedings of the 6th ACM conference on Embedded network sensor systems, pages 15–28. ACM, 2008.

[20] J.-P. Vasseur and A. Dunkels. *Interconnecting smart objects with ip: The next internet*. Morgan Kaufmann, 2010.

[21] Z. Shelby and C. Bormann. *6LoWPAN: the wireless embedded internet*, volume 43. John Wiley & Sons, 2011.

[22] A. Castellani, S. Loreto, A. Rahman, T. Fossati, and E. Dijk. *Best Practices for HTTP-CoAP Mapping Implementation*, 2012. [Online; accessed 3 November 2012]. Available from: http://tools.ietf.org/html/draft-castellani-core-http-mapping-05.

[23] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester. *Facilitating sensor deployment, discovery and resource access using embedded web services*. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, pages 717–724. IEEE, 2012.

[24] K. Hartke. *Observing Resources in CoAP*, 2012. [Online; accessed 3 November 2012]. Available from: http://tools.ietf.org/html/draft-ietf-core-observe-06.

[25] G. K. Teklemariam, J. Hoebeke, I. Moerman, and P. Demeester. *Facilitating the creation of IoT applications through conditional observations in CoAP*. EURASIP Journal on Wireless Communications and Networking, 2013(1):1–19, 2013.

[26] Z. Shelby and C. Chauvenet. *The IPSO Application Framework*, 2012. [Online; accessed 3 November 2012]. Available from: http://www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf.

[27] B. Abdulrazak, B. Chikhaoui, C. Gouin-Vallerand, and B. Fraikin. *A standard ontology for smart spaces*. International Journal of Web and Grid Services, 6(3):244–268, 2010.

[28] H. Yahyaoui, Z. Maamar, and K. Boukadi. *A framework to coordinate web services in composition scenarios*. International Journal of Web and Grid Services, 6(2):95–123, 2010.

[29] L. Gao, S. D. Urban, and J. Ramachandran. *A survey of transactional issues for web service composition and recovery*. International Journal of Web and Grid Services, 7(4):331–356, 2011.

[30] W. Kim. *Cloud computing adoption*. International Journal of Web and Grid Services, 7(3):225–245, 2011.

[31] J. M. Rodriguez, A. Zunino, and M. Campo. *Introducing mobile devices into grid systems: a survey*. International Journal of Web and Grid Services, 7(1):1–40, 2011.

[32] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. *The Click modular router*. ACM Transactions on Computer Systems (TOCS), 18(3):263–297, 2000.

[33] E. De Poorter, E. Troubleyn, I. Moerman, and P. Demeester. *IDRA: A flexible system architecture for next generation wireless sensor networks*. Wireless Networks, 17(6):1423–1440, 2011.

[34] C. Perkins, E. Belding-Royer, and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561 (Experimental), July 2003. Available from: http://www.ietf.org/rfc/rfc3561.txt.

[35] P. Hurni and T. Braun. *An energy-efficient broadcasting scheme for unsynchronized wireless sensor MAC protocols*. In Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on, pages 39–46. IEEE, 2010.

[36] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester. *The w-iLab. t testbed*. In Testbeds and Research Infrastructures. Development of Networks and Communities, pages 145–154. Springer, 2011.

[37] C. Reed, M. Botts, J. Davidson, and G. Percivall. *Ogc® sensor web enablement: overview and high level achhitecture.* In Autotestcon, 2007 IEEE, pages 372–380. IEEE, 2007.

[38] S. Jirka, A. Bröring, and C. Stasch. *Discovery mechanisms for the sensor web*. Sensors, 9(4):2661–2681, 2009.

[39] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth, et al. *SPITFIRE: toward a semantic web of things*. Communications Magazine, IEEE, 49(11):40–48, 2011.

[40] T. Foerster, D. Nüst, A. Bröring, and S. Jirka. *Discovering the sensor web through mobile applications*. Springer, 2012.

[41] Å. Östmark, J. Eliasson, P. Lindgren, A. v. Halteren, and L. Meppelink. *An infrastructure for service oriented sensor networks*. Journal of Computers, 1(5):20–29, 2006.

[42] J. Schneider, A. Klein, C. Mannweiler, and H. Schotten. *An efficient architecture for the integration of sensor and actuator networks into the future internet*. Advances in Radio Science, 9(16):231–235, 2011.

[43] L. Schor, P. Sommer, and R. Wattenhofer. *Towards a zero-configuration wireless sensor network architecture for smart buildings*. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pages 31–36. ACM, 2009.

[44] Z. Shelby, M. Koster, C. Bormann, and P. van der Stok. *CoRE Resource Directory*, 2015. [Online; accessed 15 September 2015]. Available from: http://tools.ietf.org/html/draft-ietf-core-resource-directory-04.

[45] A. J. Jara, P. Martinez-Julia, and A. Skarmeta. *Light-weight multicast DNS and DNS-SD (lmDNS-SD): IPv6-based resource and service discovery for the Web of Things*. In innovative mobile and internet services in ubiquitous computing (IMIS), 2012 sixth international conference on, pages 731–738. IEEE, 2012.

[46] B. Djamaa and M. Richardson. *Towards Scalable DNS-Based Service Discovery for the Internet of Things*. In Ubiquitous Computing and Ambient Intelligence. Personalisation and User Adapted Services, pages 432–435. Springer, 2014.

[47] B. C. Villaverde, R. De Paz Alberola, A. J. Jara, S. Fedor, S. K. Das, and D. Pesch. *Service discovery protocols for constrained machine-to-machine communications*. Communications Surveys & Tutorials, IEEE, 16(1):41–60, 2014.

3

Flexible Unicast-Based Group Communication for CoAP-Enabled Devices

"The single biggest problem in communication is the illusion that it has taken place."

– George Bernard Shaw (1856 - 1950)

The discovery of CoAP servers and their resources, as discussed in Chapter 2, is only a first step towards the realization of IoT applications. To further facilitate the creation of IoT applications, there is a need for novel enablers. An example of such a need is the ability to simultaneously interact with a multitude of constrained devices or their resources at the same time. To address this need, we provide a solution in this Chapter for building groups of resources. The CoAP Internet draft and the CoAP group communication Internet draft – which addresses the same need – were almost stable at the time of writing this chapter. Also an open source implementation of CoAP for the Contiki OS was available. An implementation of multicast was available, but not part of the Contiki OS yet.

⋆ ⋆ ⋆

I. Ishaq, J. Hoebeke, F. Van den Abeele, J. Rossey, I. Moerman, and P. Demeester.

Published in Sensors Magazine, 4 June 2014.

Abstract Smart embedded objects will become an important part of what is called the Internet of Things. Applications often require concurrent interactions with several of these objects and their resources. Existing solutions have several limitations in terms of reliability, flexibility and manageability of such groups of objects. To overcome these limitations we propose an intermediately level of intelligence to easily manipulate a group of resources across multiple smart objects, building upon the Constrained Application Protocol (CoAP). We describe the design of our solution to create and manipulate a group of CoAP resources using a single client request. Furthermore we introduce the concept of profiles for the created groups. The use of profiles allows the client to specify in more detail how the group should behave. We have implemented our solution and demonstrate that it covers the complete group life-cycle, *i.e.*, creation, validation, flexible usage and deletion. Finally, we quantitatively analyze the performance of our solution and compare it against multicast-based CoAP group communication. The results show that our solution improves reliability and flexibility with a trade-off in increased communication overhead.

3.1 Introduction

The Do-It-Yourself (DIY) movement is spreading beyond traditional domains, such as home painting, to more modern domains, such as programming. DIY programming gets especially interesting when it involves real-time data from the growing amount of smart objects with embedded sensors and when actuators can be triggered to perform real-world actions accordingly. It becomes even more interesting and appealing when access to these smart objects can be obtained over the ubiquitous Internet—leading to what is now mostly known as the Internet of Things (IoT). However, these smart objects are typically optimized for low-power consumption and low-cost. They are constrained in their processing capabilities (CPU, RAM, ROM,...) and thus unable to run standard Internet protocols. The networks that connect these objects together are often referred to as Low-power and Lossy Networks (LLNs).

Connecting LLNs to the Internet, communicating with smart objects, and manipulation of sensor data and actuators was largely done in proprietary ways. Each vendor had its own set of protocols and tools to access, interpret and if needed manipulate sensor data and to trigger actuators. More recently a lot of effort has been put into the development of open standards that cover many aspects of communication and access to smart objects. At the networking layer 6LoWPAN allows IPv6 communication with these objects through an adaptation layer [1]. At the application layer standards are being prepared to allow access to these objects in a

RESTful way, similar to how most information on today's Internet is accessed over HTTP. The main driver behind this is the Internet Engineering Task Force (IETF). The IETF established the Constrained RESTful Environments (CoRE) working group with the aim of realizing the REST architecture in a suitable form for the most constrained nodes and networks. Constrained devices are turned into embedded web servers that make their resources accessible via the CoAP protocol. CoRE is aimed at Machine-to-Machine (M2M) applications such as smart energy and building automation [2].

Typically, each of the constrained servers has at least one CoAP resource that may be queried by clients to obtain information about the smart objects themselves (e.g., battery level), about the environment that they monitor (e.g., temperature of the room), or to trigger the objects to perform real-world actions (switch the light on). These CoAP resources are identified by a Uniform Resource Identifier (URI) such as coap://[aaaa::1]/temperature.

Depending on the application, information from individual objects might not be sufficient, reliable, or useful. An application may need to aggregate and/or compare data from several nodes in order to obtain accurate results. In the same way, a single user request might need to trigger a series of actions on multiple actuators. This need to communicate with groups of objects is obvious in many IoT scenarios. For example, in a smart home, when you leave your bed during the night, you might want that the lights in the bedroom, hall and toilet turn on automatically until you go to bed again. Or, when suspicious movement is detected in the living area during the night, several actuators may be triggered such as an alarm going off and particular lights being turned on or made flashing. From these two simple examples, one can already see that the same lights can be parts of different groups according to the needs of the user. The needs can change regularly and thus the grouping and ungrouping of resources should be flexible and easy. Similar examples for the need of group communication can be found in virtually any IoT scenario.

The need for group communication is very well recognized in the IETF. This can be clearly seen from the charter of the IETF CoRE Working Group. The charter clearly states that the "initial work item of the WG is to define a protocol specification for CoAP that includes ... the ability to support a non-reliable multicast message to be sent to a group of Devices to manipulate a resource on all the Devices in the group." The charter also states that "the working group will not develop a reliable multicast solution" [3].

Although multicast may be used to transmit the same request to several objects, multicast communication in LLNs has some disadvantages. For instance, it is more difficult to route multicast traffic with a minimum of message duplication at the receiving hosts than in the case of unicast. Furthermore, basic multicast is not reliable in an LLN, which is problematic for requests that require guaranteed

delivery. Also, the creation of multicast groups, defining which objects should be addressed when using a particular multicast address, is hard to realize inside LLNs. Additionally, the use of network wide multicast increases the footprint of the code that needs to fit on the constrained objects, and it is to be expected that this functionality will not be available in many LLNs.

As an alternative, unicast-based solutions may be considered. Some unicast-based solutions (such as reliable messaging) have been introduced to alleviate some of the problems above, but these features are insufficient. The current CoRE drafts do not foresee any unicast-based way to manipulate resources that are located on multiple smart objects with a single client request. To overcome this shortcoming and be able to perform such composite requests, intelligence is typically added to the client application to make it communicate with the smart objects individually. This leads to more complex user applications, and the added intelligence and programming cannot be easily shared with other applications. Furthermore, complex user applications may be unmanageable. Any modifications to those complex user applications may require significant testing time, thus limiting the flexibility of the user applications. Additionally a large overhead of communication between the client machine and the smart objects is generated, especially when many smart objects are involved in these actions. When the communication between the client and the smart objects is done across the Internet, delays are unpredictable and a sequence of actuator commands might arrive out of order and possibly have unwanted results. Furthermore, if the communication occurs over costly links, communication between the client and the smart objects might get unnecessarily expensive.

In this paper we propose a novel solution for communication with a group of resources across multiple smart objects based on CoAP unicast. The group members can be homogeneous or heterogeneous, on a single node or on multiple nodes, or another group. The group that we create is itself exposed as a RESTful CoAP resource, and thus can be accessed by any CoAP client (including other constrained devices). We include optional validation of the group at creation time; we attach a profile to the created group and thus can customize its behavior and provide fine-grained control over it. We have implemented our solution and provide a functional and performance evaluation for it. In the past, we have already presented our concept along with an initial implementation in [4]. In this expanded article, we elaborate on the concept, add more advanced features to the implementation, compare our solution with multicast and evaluate its functionality and performance.

The remainder of this paper is structured as follows: first, we will briefly provide an overview of CoAP in Section 3.2. We then discuss CoAP group communication requirements and related work in Sections 3.3 and 3.4. Next, in Section 3.5, we describe our approach in detail. In Section 3.6, we present our implementation

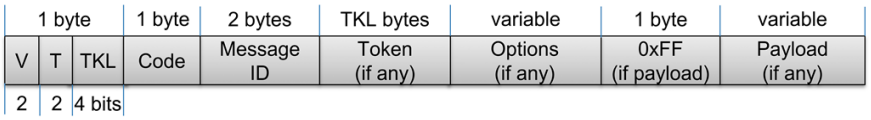


Figure 3.1: CoAP Message Format consisting of a 4-bytes base binary header followed by optional extensions.

and evaluate the functionally and the performance of our solution. In Section 3.7, we discuss the results and compare them to the requirements. Section 3.8 concludes this work with a summary and outlook.

3.2 CoAP overview

The focus of this paper is to enable interaction with a group of devices from a service/application perspective in a way that is in line with ongoing standardization activities in the field of IoT. In the last few years a lot of effort has been put in defining a standard application protocol, similar to HTTP, but more suitable for constrained devices, namely CoAP. The base CoAP protocol is defined in draft-ietf-core-coap [5] in conjunction with a number of additional specifications. In this section we briefly introduce the base CoAP specification and those extensions that are relevant to our group communication work.

3.2.1 Base CoAP

CoAP uses the same RESTful principles as HTTP, but it is much lighter so that it can run on constrained devices [6, 7]. To achieve this, CoAP has a much lower header overhead and parsing complexity than HTTP. It uses a 4-bytes base binary header that may be followed by compact binary options and payload. Figure 3.1 shows the CoAP message format as specified in version 18 of the draft. This version was approved by the Internet Engineering Steering Group (IESG) in July 2013 and was at the time of writing this article being edited by the RFC editor to convert the draft into an RFC. Thus, it is expected that this will be the final CoAP message format.

The CoAP interaction model is similar to the client/server model of HTTP. A client can send a CoAP request, requesting an action specified by a method code (GET, PUT, POST or DELETE) on a resource (identified by a URI) on a server. The CoAP server processes the request and sends back a response containing a response code and payload. Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport layer such as UDP and thus also supports multicast requests. This allows CoAP to be used for point-to-multipoint interactions which are commonly required in automation. Optional reliability is

supported within CoAP itself by using a simple stop-and-wait reliability mechanism upon request. Secure communication is also supported through the optional use of Datagram Transport Layer Security (DTLS) [8]. As can be seen in Figure 3.1 all CoAP messages start with a 4-bytes base binary header that consists of the following fields:

- *Version (V)*: indicates the CoAP version number. Current version is 1.
- *Type (T)*: indicates if this message is of type Confirmable, Non-Confirmable, Acknowledgement or Reset.
- *Token Length (TKL)*: indicates the length of the variable-length Token field.
- *Code*: indicates if the message carries a request (1–31), a response (64–191), or is empty (0). In case of a request, the Code field indicates the Request Method (GET, POST, PUT and DELETE); in case of a response a Response Code.
- *Message ID*: is used for the detection of message duplication, and to match messages of type Acknowledgement/Reset to messages of type Confirmable/ Non-confirmable.

To be able to offer communication needs that cannot be satisfied by the base binary header alone, the base 4-bytes header may be followed by one or more of the following optional fields:

- *Token*: the Token is used to correlate requests and responses.
- *Options*: an Option can be followed by the end of the message, by another Option, or by the Payload Marker and the payload.
- *Payload*: if present and of non-zero length, it is prefixed by a fixed, one-byte Payload Marker (0xFF) which indicates the end of options and the start of the payload. The payload data extends from after the marker to the end of the UDP datagram, *i.e.*, the Payload Length is calculated from the datagram size. The absence of the Payload Marker denotes a zero-length payload.

CoAP defines a number of options which can be included in a message. Both requests and responses may include a list of one or more options. Each option instance in a message specifies the Option Number, the Option Length and the Option Value of the defined CoAP option. As an example of a simple CoAP option consider the Content-Format option. This option indicates the representation format of the message payload. This option has the Option Number 12 and its Option Length is between zero and two bytes. The Option Value itself is a numeric

content format identifier that is defined in the CoAP Content Format Registry (Section 12.3 of the draft [5]). Another example is the Max-Age option which has the Option Number 14. This option indicates the maximum time a response may be cached before it is considered not fresh. The Option Value is an integer number of seconds between 0 and inclusive (about 136 years). If this option is not included in any CoAP response, it can be assumed that the response will be fresh for 60 s and thus will not be queried again by a cache within this time frame.

When using confirmable messages CoAP tries to achieve reliability by using a simple stop-and-wait retransmission with exponential back-off. By default the initial back-off is set to a random time between 2 and 3 s. This means that if a reply to a confirmable packet is not received within the initial back-off time, the CoAP sender will double the initial back-off time and retransmit the packet. If a reply to the first retransmission is not received, CoAP will again double the back-off time and retry the transmission until MAX_RETRANSMIT (by default 4) is reached. If no reply is received after expiring of the back-off time of the last retransmission, the client will be notified about the error condition.

The IETF CoRE working group considers constrained RESTful environments as an extension of the current web architecture. The group envisions that CoAP will complement HTTP and that CoAP will be used not only between constrained devices and between servers and devices in the constrained environment, but also between servers and devices across the Internet [9]. An important requirement of the CoRE working group is to ensure a simple mapping between HTTP and CoAP so that the protocols can be proxied transparently. Thus proxies and/or gateways play a central role in the constrained environments architecture. These proxies have to be able to communicate between the Internet protocol stack and the constrained environments protocol stack and to translate between them as needed.

3.2.2 Resource discovery

In M2M applications where there are no humans in the loop, it is important to provide a way to discover resources offered by a constrained server. For HTTP Web Servers, the discovery of resources is typically called Web Linking [10]. The use of Web Linking for the description and discovery of resources hosted by constrained web servers (CoAP or HTTP) is specified by the CoRE Link Format – RFC 6690 [11]. A well-known relative URI "/.well-known/core" is defined as a default entry-point for requesting a list of links to resources hosted by a server. Once the list of available resources is obtained from the server, the client can send further requests to obtain the value of a certain resource. The example in Figure 3.2 shows a client requesting the list of the available resources on the server (GET /.well-known/core). The returned list (in CoRE Link Format) shows that the server has, amongst others, a resource called /s/t that, when queried, returns the

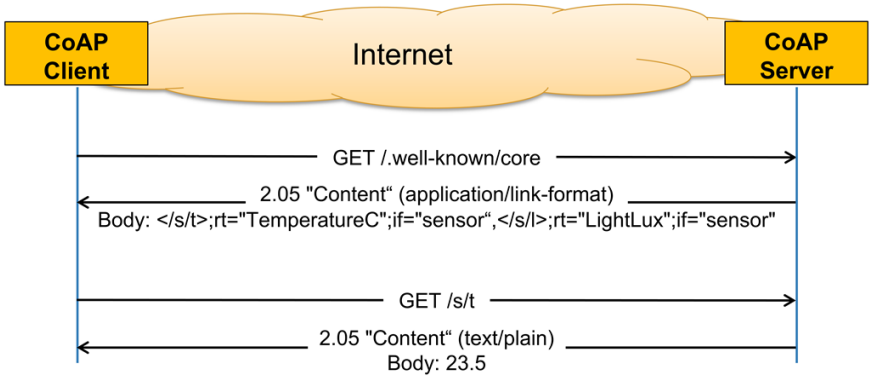


Figure 3.2: An example of Constrained RESTful Environments (CoRE) direct resource discovery and Constrained Application Protocol (CoAP) request.

temperature in degrees Celsius. The client then requests the value of this resource (GET /s/t) and receives a plain text reply from the server with the value of the current temperature as payload of the message (23.5).

However in many M2M scenarios, nodes might have long sleeping periods and thus making direct discovery of resources not practical. To solve this problem, the CoAP community is proposing to use CoRE Resource Directories (RD) that host descriptions of resources held on other servers [12]. This way a CoAP server can register its resources with one or more RDs. Clients in turn can discover these resources by performing lookups against an RD. For example the same resource discovery that was performed by using direct communication between the client and the server in Figure 3.2 can now be performed by using an RD as illustrated in Figure 3.3. For more details about the registration and lookup interfaces of Resource Directories we refer to [12].

3.2.3 Blockwise transfer

In many cases the payloads that CoAP needs to carry are very small (e.g., just a few bytes for temperature sensor, door lock status or toggling a light switch). In these cases the basic CoAP message provides very efficient means of communication and works very well. However in some cases CoAP needs to handle larger payloads (e.g., images or firmware update). Since CoAP is based on datagram transports such as UDP or DTLS, data fragmentation and reassembly is not offered by these transport protocols. Relying on IP fragmentation is also not very helpful, because IP fragmentation can handle only payloads up to 64 KB. Thus, providing a mechanism at the application layer that is able of transferring large amounts of data in smaller pieces becomes a necessity. This will not just help avoiding the 64

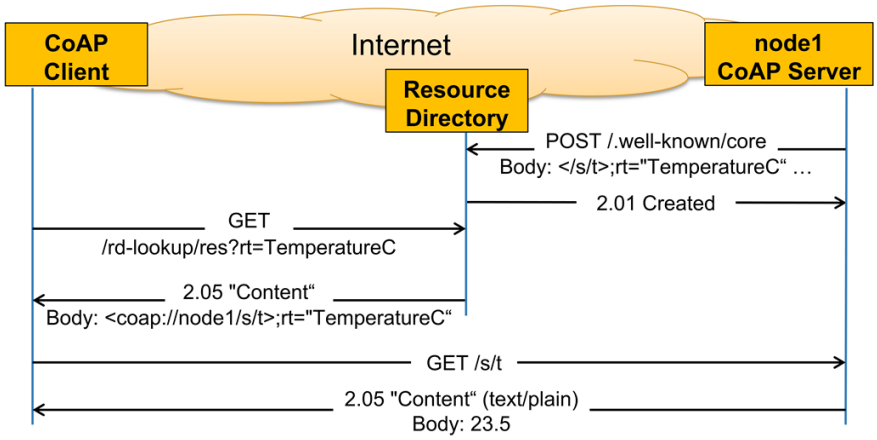


Figure 3.3: An example resource discovery by using a Resource Directory.

KB UDP datagram limit, but will also help avoiding both IP fragmentation (MTU of 1280 for IPv6) and also 6LoWPAN adaptation layer fragmentation in LLNs (60–80 bytes).

To overcome the payload size limitation, draft-ietf-core-block defines two CoAP options: Block1 and Block2 [13]. By using this pair of options CoAP becomes capable of transferring a large payload in multiple smaller CoAP messages. Both Block1 and Block2 options can be present both in request and response messages. In either case, the Block1 Option pertains to the request payload, and the Block2 Option pertains to the response payload. Block sizes are represented inside the Block1 and Block2 Options as a three-bit unsigned integer called indicating the size of a block to the power of two. Thus:

$$blocksize = 2^{SZX+4} \tag{3.1}$$

The allowed values of SZX are 0 to 6 and thus the resulting allowed block sizes are: 16, 32, 64, 128, 256, 512 and 1024 bytes.

6LoWPAN might start using fragmentation/reassembly for datagrams as soon as the payload size gets larger than 60 bytes. This fragmentation/reassembly process burdens the lower layers with conversation state and is sometimes not implemented to conserve resources at the constrained devices. To avoid such fragmentation and reassembly, blockwise transfer with Block1 and Block2 sizes of 16 or 32 should be used whenever the payload exceeds 60 bytes.

An important aspect of the blockwise transfer mechanism is that often the server can handle block transfers in a stateless fashion. It does not require connection setup and the server does not need to track each transfer separately and thus conserves memory.

3.2.4 Group communication

The IETF CoRE working group has recognized the need to support a non-reliable multicast message to be sent to a group of devices to manipulate a resource on all the devices in the group. Therefore, they have developed the "Group Communication for CoAP" Internet Draft [14], which provides guidance for how the CoAP protocol should be used in a group communication context. *Group Communication* refers to sending a single CoAP message to all members of a specific group by utilizing UDP/IP multicast for the requests, and unicast UDP/IP for the responses (if any). This implies that all the group members (the destination nodes) receive the exact same message. The solution proposed by the IETF CoRE working group is discussed further in Section 3.4.

3.3 Group communication requirements

In our work we broaden the *CoAP group communication* definition from Section 3.2.4: CoAP-based group communication is a *method to manipulate a group of resources on devices using CoAP as the underlying protocol*. Such a group of resources is called an *entity* and the resources themselves are called the entity *members*. We classify two types of entities based on the entity members:

- *Homogeneous Entity*: is an entity in which the members share a common set of properties (URI path, method, content-type, block-size, observe, *etc*).
- *Heterogeneous Entity*: is an entity in which not all members share a common set of properties.

Within this context, we now define the requirements and goals for CoAP group communication and motivate their importance in the context of IoT applications, constrained devices and LLNs:

1. *Flexibility*: as it is expected that the IoT will contain a huge amount of devices, it is also expected that the amount of device types will be enormous. To interact with a subset of these devices in a group, the group communication solution should be very flexible to accommodate the differences between devices and device types. In particular the solutions should be flexible enough to offer:

   (a) *Support for homogeneous and heterogeneous groups*. CoAP servers may be heterogeneous in terms of their CoAP resources, even if they provide the same functionality. For example the IPSO (Internet Protocol for Smart Objects) Alliance has published an Application Framework that recommends a classification of resources based on their functionality by defining a set of Resource Types [15]. However even if a

group of resources offers the same functionality (same resource type), the actual resource path of the resource on different group members might be different. In some cases one might even want to have a group of resources with different resource types and simply query it (e.g., collecting heterogeneous environmental data). Also other types of heterogeneity in the resources are possible: e.g., different payload to PUT request, different media-types, *etc*.

(b) *Support of group members that are not part of the same network*. Often, it is assumed that all members in a group belong to the same network. However, group communication solutions should not be limited to this setting. In the future, it may as well be that group communication involves nodes from different sensor networks, networks that may be co-located or spread over different locations.

2. *Light-Weight (footprint)*: the group communication solution should have limited footprint on constrained devices. It is expected that a lot of the IoT devices will be of Class 1 (˜10 KB of RAM, and ˜100 KB of ROM) [16]. Any overhead involved by a group communication solution should not prevent the solution from running on Class 1 devices. Furthermore the solution should scale with the number of groups a certain member can be part of.

3. *Use of Standards*: to allow the creation of groups across a variety of members from different vendors and domains, it is mandatory to use standard protocols that are widely supported. The focus of our work is on using CoAP as an application layer standard protocol. As mentioned in Section 3.2, CoAP consists of a base protocol and a set of optional extensions. It is expected that not all CoAP servers will support all CoAP extensions. Thus it becomes essential to limit the use of optional extensions in order not to exclude potential CoAP servers of becoming group members due to missing extensions.

4. *Performance*: CoAP is designed to run on resource constrained devices. In order to keep it this way, any CoAP group communication solution should have little overhead and be efficient in the use of resources of the nodes and the LLN. In particular the number and size of messages sent in the Low-power and Lossy Network (LLN) should be kept to a minimum, in order to conserve valuable node energy (nodes are often battery powered, or harvest energy from the environment). A very powerful method for limiting the number of messages inside the LLN is using efficient caching techniques. This not only limits the number of messages and energy consumption, but it also decreases response latency. CoAP transactions and options are thus well optimized to support caching whenever possible. Group communication should not be an exception and should not hinder the use of caches.

5. *Validation and Error Handling*: since a group might include heterogeneous members, it should be possible to validate the group in order to make sure that the group works as intended. The group should have mechanisms for reporting and handling error conditions such as node or route failures.

6. *Reliability*: sometimes it is not essential to get reliable replies from all group members (e.g., it might be enough to get the temperature measurements from just one of the many temperature sensors in a room). However in many other cases, it can be important to have reliable communication with all group members. For example, one would expect that all lights in the room would go on when one flips on the light switch.

7. *Ease of Group Manipulation*: the needs of the user might change with time and thus group membership might also change. In dynamic environments the changes might be frequent. It is important to be able to handle such changes easily. One should avoid node reconfigurations, as this might be a tedious task. Also it should be possible for nodes to be part of different groups at the same time or at different times.

8. *Expressiveness/Control*: there are several results that one might want to achieve by interacting with a group of objects/object resources. In some cases one might be interested in all the individual results of all members as in the case of turning the lights on. In many other cases the individual values might not be of interest at all. In these cases one might be interested in an aggregated value (e.g., min, max, avg,...) of all, or even of just a subset of, the group members. Thus it is desired to have support for processing of individual group member results and replying to the requester with aggregated results. For example it should be possible to query a certain subset of the members and compute the average, or reliably update all members.

9. *Security*: secure communication might be of little interest inside a shielded and controlled environment. However, by exposing sensors and actuators to the Internet, security becomes a major concern. In some scenarios having an end-to-end security is a strict requirement. Communicating with a group of resources is no exception. In fact it is even more sensitive than communicating with an individual resource, since compromising the group means compromising all the individual members.

3.4 Existing solutions

As mentioned, to address the group communication needs, the IETF CoRE Working Group has developed the "Group Communication for CoAP" Internet Draft [14]. This draft discusses fundamentals and use cases for group communication

patterns with CoAP and provides guidance for how the CoAP protocol should be used in a group communication context. The draft provides an approach for using CoAP on top of non-reliable IP multicast and does not attempt to provide a reliable solution for CoAP group communication as set forward by the Working Group charter. Certainly the use of multicasts allows reducing the amount of requests in the LLN, by sending one request to several destinations at the same time. However, multicasts are not cache-friendly, preventing possible reduction of requests and replies by utilizing caches. Depending on the use case and network topology, the reduction of packets as a result of using a cache can be better than the reduction obtained from using multicasts. This approach exhibits the limitations of multicasts as discussed in Section 3.1. Also multicasts are not useful when a single user action needs to trigger different sensor requests, since one multicast request delivers the same message to all group members. Additionally, secure communication with the group members is not possible, since all communication based on this draft operates in CoAP NoSec (No Security) mode. Finally, multicast is not supported on all LLN Media Access Control (MAC) protocols, especially MAC protocols that use Radio Duty Cycles (RDC) to shut down their radios when not in use. For example Xmac does not support multicast since it shuts down its receiver to avoid overhearing. Special MAC protocols that support multicast have been proposed such as in [17]. Interestingly, this MAC protocol will send the multicast data to each receiver one by one (unicast) if the multicast data drops below a certain threshold.

As mentioned, the use of multicast as a means to interact with multiple objects concurrently requires multicast support in the network. Typically IP multicast relies on topology maintenance mechanisms to discover and maintain routes to all subscribers of a multicast group. However, maintaining such topologies in LLNs may not be feasible given the available resources. As a result, special multicast protocols have been proposed for the use inside LLNs. For example, the "Multicast Protocol for Low power and Lossy Networks (MPL)" internet draft uses the Trickle algorithm to manage message transmissions for both control and data-plane messages and avoids the need to construct or maintain any multicast forwarding topology [18]. An alternative is the stateless multicast RPL forwarding (SMRF) algorithm, which according to [19] achieves significant delay and energy efficiency improvements at the cost of a small increase in packet loss. Regardless of the used multicast protocol, all nodes on the path between the sender and receivers must be extended to support the protocol.

The "Group Communication for CoAP" Internet Draft was the basis for the work presented in [20], in which web services based CoAP multicasts were used to access data from Building Automation Systems (BAS). It shows how using multicasts allows creating basic building control scenarios without the need of a central control unit. Certainly this approach has several advantages such as eliminating

the need for a control unit, often less power consumption than using unicasts and its suitability in many non-critical use cases (due to the lack of reliability of multicasts). However this approach exhibits the limitations of multicasts as discussed in this section above.

Simple unicast solutions are defined in the CoRE Interfaces draft [21]. Among other interface types, this draft defines the Batch interface type and its extension, the Linked Batch interface type. Batch interfaces are used to manipulate a collection of sub-resources at the same time. Contrary to the basic Batch, which is a collection statically defined by the web server, a Linked Batch is dynamically controlled by a web client. The resources forming the linked batch are referenced using Web Linking [10] and the CoRE Link Format [11]. The draft does not foresee any way to manipulate resources that are located on multiple smart objects with a single client request.

An approach somewhat more similar to ours, also using the notion of an entity, has been presented in [22]. The aim here is to annotate real-world objects by using entities that are automatically created based on semantic information, which resides on the constrained devices. One problem of using semantics on constrained devices is that semantics can easily require a lot of memory that might not be available on the constrained devices. Further, in our approach users can create entities as required and we address important aspects related to entity validation and entity behavior.

The authors of [23] present an extension to CoAP called SeaHttp, that enables communication with a group of resources. Similar to our work, SeaHttp also uses unicasts to realize group communication. The authors propose to extend CoAP with two additional methods (BRANCH and COMBINE) to allow members to join and leave groups without the need for a separate group manager. This means that members should have the intelligence to know which group they should join/leave. Constrained devices will not have this intelligence, so again, a "manager" will be needed to inform the devices so they can take appropriate actions. Furthermore, BRANCH and COMBINE can maybe reduce the number of messages; however the trade-off is the need to implement a new mechanism. It is better to use an approach that can be plugged in into any existing network without major modifications (or at least not a modification to every node). The article does not discuss if the use of caches will still be possible with SeaHttp resources. However, should this be possible then also the caches should be extended accordingly. Finally this approach does not have the flexibility we target, since group members have to be reprogrammed with the groups they should join each time the requirements of the user changes.

To our knowledge, these are the only works that explore communication solutions for interacting with a group of CoAP-enabled constrained devices. Next to these, there exist other solutions to realize or improve multicast communication in

Wireless Sensor Networks, such as [18, 24]. These solutions can alleviate some of the problems related to (reliable) multicasting, but their scope is different from the work presented here.

3.5 Group communication using unicasts

We aim to create an intermediate level of aggregation to be able to easily manipulate a group of resources across multiple smart objects. To avoid increasing the *footprint* of the constrained devices, we use the *same technology* as used to manipulate individual resources, *i.e.*, CoAP, and extend it accordingly. Such a group of resources is called an *entity* and the entity can be used or manipulated through a single CoAP request. Similarly, the creation of an entity by a client is realized via a single CoAP request and includes a complete *validation* of the entity. Furthermore we introduce the notion of profiles for the created entities. The use of entity profiles allows the client to specify in more detail how the *entity should behave* (e.g., if it should use confirmable or non-confirmable CoAP messages), and, through updating the profile, allows manipulation of this behavior. As such, we strive to combine ease of creation, ease of usage and *flexibility* in behavior into a complete solution for interacting with CoAP resources from different objects inside a LLN. By building upon *standardized* concepts, the impact on the constrained devices is limited. In the following subsections we discuss the details of our approach.

3.5.1 System overview

We call the component that manages the entities, Entity Manager (EM). This component, which can reside, e.g., on the Border Gateway of the LLN, is responsible for maintaining entities that are created from groups of resources residing on CoAP servers (*i.e.*, sensors and actuators) inside the LLN. Clients on the Internet can interact with an EM to create new entities and/or customize how these entities should behave. Optionally the client can elect to contact a resource directory [12] in order to discover which resources are available in the network. Figure 3.4 shows an overview of the involved components.

The EM functionality does not have to be put on a dedicated device. Theoretically any CoAP server can be extended to become an EM (Figure 3.5). The choice of the most appropriate location to put the EM functionality depends on the size and topology of the network. For example, it can reside on a smart object in the constrained network with enough resources, in the Cloud, on the client device itself, or on a gateway at the edge of the LLN. The latter case has the added benefit that security can be centrally managed besides offloading the processing from constrained devices. One can also decide to implement multiple EMs (at the same or at different locations) to avoid having a single point of failure and thus improving

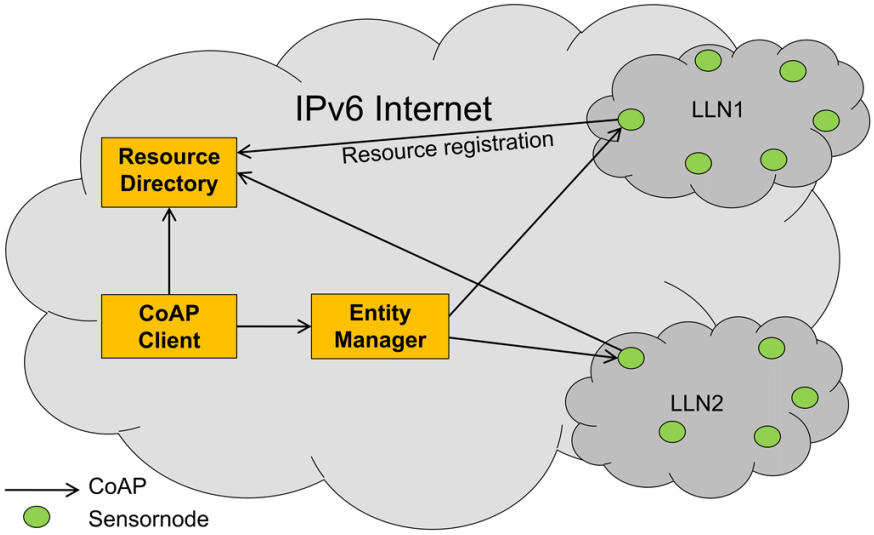


Figure 3.4: Clients create entities consisting of several smart object resources on the EM. Clients can optionally query a resource directory to discover the existence of the resources.

reliability, availability and scalability.

Regardless of the location of the EM, it will serve as a "proxy" between the client and the constrained devices. Client requests will be sent to the EM, which will analyze and verify the requests and then issue the appropriate requests to the constrained devices using CoAP. Once the EM receives responses from the constrained devices, it will combine them according to the needs of the client and will send back an aggregated response to the client.

When a client tries to create a new entity consisting of a group of resources inside LLNs, the EM performs a sanity check on the request in order to make sure that the resulting entity would make sense. For example it verifies that the resources inside the entity are valid, whether they support a certain content format and whether their data can be aggregated. Customization of the entity behavior is accomplished by creating profiles for the entities. A profile of an entity can specify for example whether to return the values of all resources in the entity, only the computed average of all values or a subset of all values. Figure 3.6 shows a high-level structure of the EM. It shows that the EM contains two databases:

- *Entity Database*: in this database all entities are stored along with their profiles as defined by the user.
- *Capabilities Database*: this optional database provides rules and knowledge that can be used to match user requests with sensor capabilities. This can be as simple as translating a request for temperature in degrees Celsius while

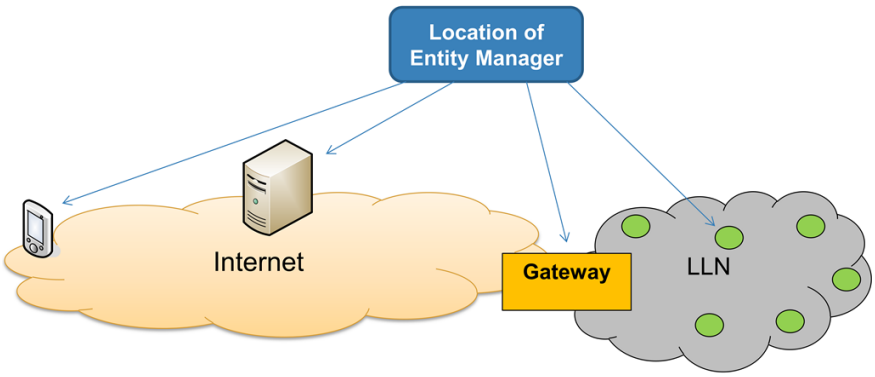


Figure 3.5: The EM functionality can be integrated into any CoAP server. The optimal location for the EM depends on the use case.

obtaining the data from a sensor that only supports Fahrenheit. It can also be more complex, e.g., converting resource representations from one content format into the other.

3.5.2 Entity creation

To facilitate the creation and manipulation of entities, the EM offers a CoAP resource "/e". We call this resource the Entity Management Resource. This interface only supports the CoAP POST request method. As payload of the request, it expects a collection of resources in CoRE link format [11], which together should form the entity. In the response, the Location-Path CoAP option is used to specify the name of the newly created resource. In the current design, the payload of the response is in plain text and describes the results of the validation tests performed by the EM on the collection of resources.

Thus, when a client wants to create an entity consisting of several members, it has to compose a CoAP POST request and send it to the Entity Management resource on the EM. The EM creates the entity, assigns it a unique URI, and stores the entity in the entity database for future usage. Then the EM starts the entity validation process (explained in the next subsection). The client is informed about the URI to use in order to access or further customize the newly created entity and about the results of the validation of the entity. In addition, the new entity resource can be registered in a resource directory as well, making it available for lookup. If the entity did not pass the validation process the client should fix any errors and resubmit the entity for validation again before the client can use the entity.

An example of the entity creation process is shown in Figure 3.7. In this simple example the client requests the creation of an entity consisting of two members:

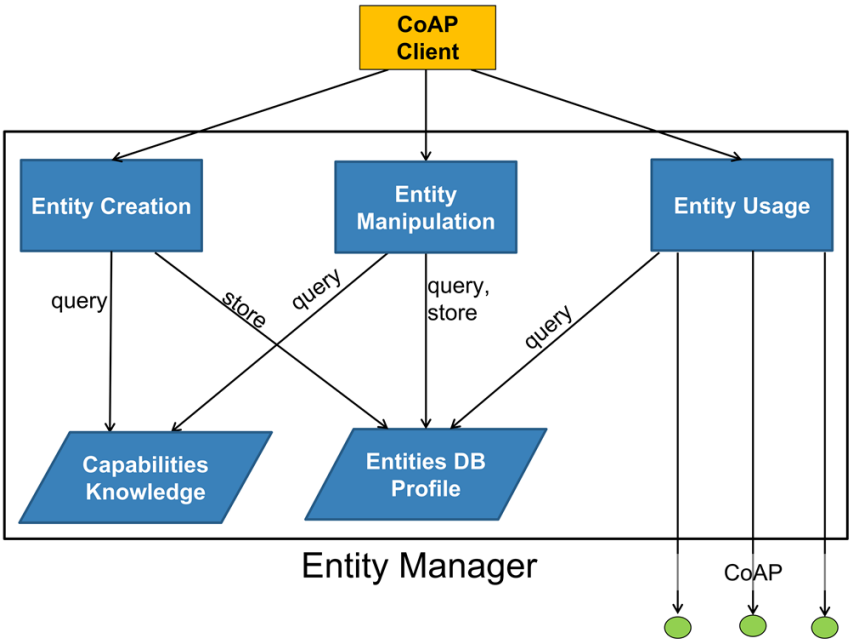


Figure 3.6: Entity Manager (EM) high-level structure.

`coap://[Sen5]/tmp` and `coap://[Sen8]/tmp`, with `Sen5` and `Sen8` the IPv6 addresses of the two sensors. The EM creates the new entity, assigns it the URI `/1` and informs the client about the newly created entity. From now on, any client can access the newly created entity by accessing the `/1` resource on the EM. Please note the validation process is not shown in Figure 3.7 for simplicity.

At creation time, the client can use optional URI-Query CoAP options with the POST request to specify the name of the entity to be created or to customize the default behavior of the entity. For example, to create the entity "room_humidity" that returns by default the minimum value of all members when queried a POST to `coap://[EM]/e?path="/room\_humidity"\&eo="min"` is needed. We will discuss customization of the entity behavior in more detail in Section 3.5.5.

3.5.3 Validation process

Whenever a client requests to create a new entity or to modify an existing entity, the EM performs a validation process. The purpose of this validation process is twofold:

1. Make sure that the members in the entity exist and can be used.

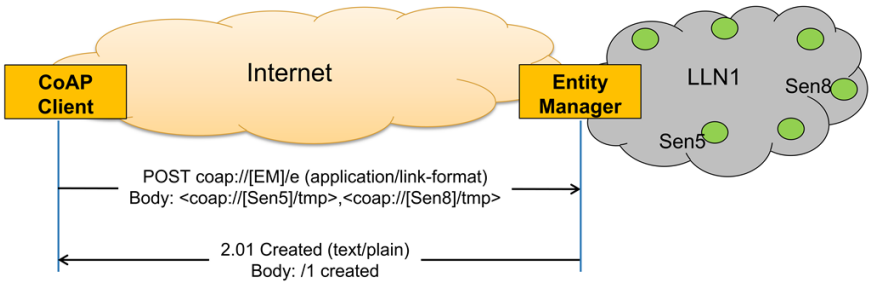


Figure 3.7: A CoAP client requesting from an EM to create a new entity that contains two resources.

2. Derive the properties of the entity based on the properties of the members it contains.

If the entity successfully passes validation the EM marks the entity as a valid entity and stores the entity along with its calculated properties in the entity database for future usage. If the entity fails validation it is still created, but marked as invalid. The entity validation is based on EM's knowledge of the individual members and their profiles and based on the knowledge in the capabilities database as will be discussed in the next paragraphs.

Resource profiles can be used to express capabilities of a CoAP server and its resources [25]. Profiles are usually expressed in JSON format [26]. To briefly illustrate resource profiles, let us assume that in Figure 3.7 the temperature sensor at `coap://[sen5]/tmp` supports the `Uri-Host` (3), `ETag` (4), `Observe` (6), `Uri-Port` (7), `Uri-Path` (11) and `Content-Format` (12) CoAP options (op). This sensor further supports the `application/json` (50) content format (cf) and the allowed method (m) is GET (1). This will result in sen5 having the following profile:

```
Res: 2.05 Content (application/json)
{
  "profile":[
    {
      "path":"tmp",
      "op":[3,4,6,7,11,12],
      "cf":[50],
      "m":[1]
    }
  ]
}
```

If the EM does not know any of the members in an entity (e.g., based on knowledge in a resource directory) or does not know the member capabilities, it tries to obtain this information according to a fallback mechanism as follows:

1. The EM tries to contact the object containing the resource in order to obtain the resource profile, since this returns the most complete information about the resource.
2. If the resource profile does not exist, the EM tries to derive information about this resource from `/.well-known/core` of the respective object.
3. If this fails as well, the EM tries to query the resource directly to discover, as a minimum, if the resource exists or not.

The validation process that the EM performs on entities is shown in a simplified form in Figure 3.8. In essence the process will:

- Verify whether the individual members contained in the entity are valid (*i.e.*, the resources exist on the respective nodes).
- Derive the operations that can be performed on the entity, based on the operations supported by the individual members (e.g., which CoAP options are supported, which RESTful methods are allowed?).
- Verify whether the individual members do not conflict. A sample conflict can occur when an entity creation request contains a sensor member that supports only the GET method and an actuator member that supports only the PUT method.
- Verify whether the responses sent by the individual members can be combined together using a common denominator or knowledge from the capabilities database.

Once the EM knows all information about the members that should become part of the entity and once all necessary checks have been passed, the EM creates a profile for the entity based on this information and the EM's capabilities database. To illustrate this let us further assume that the second temperature sensor in Figure 3.7 `coap://[sen8]/tmp` supports the same options as sen5 except for the observe option. Only the GET method is allowed and the supported content formats on this sensor are `text/plain` (0) and `application/json` (50). Thus sen8 will have the following profile:

```
Res: 2.05 Content (application/json)
{
  "profile":[
```

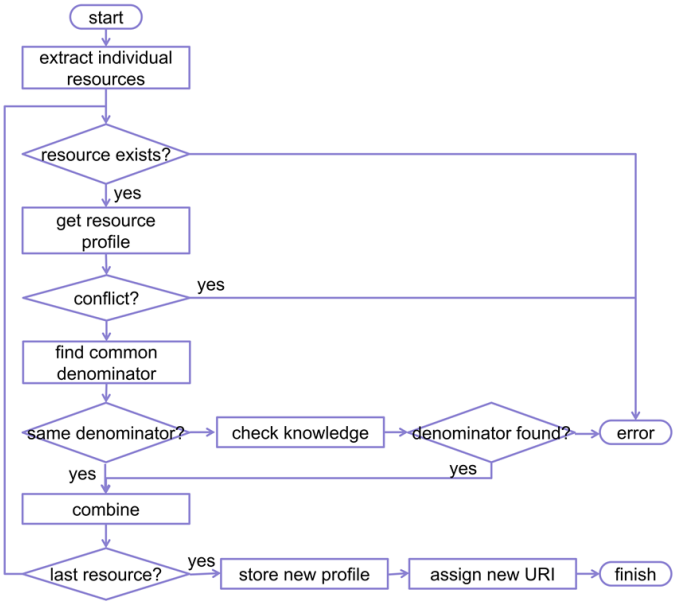


Figure 3.8: Entity validation process flow.

```
    {
      "path":"tmp",
      "op":[3,4,7,11,12],
      "cf":[0,50],
      "m":[1]
    }
  ]
}
```

Based on these two profiles the EM constructs a profile for the newly created entity. This profile contains information related to the resource itself, as described in [25]. In this example, this includes the options that are supported, the supported methods (only GET) and the content format `application/json` (50). In addition, the profile is extended with an entity specific part, providing more information about the entity itself. The resulting profile of the entity looks as follows:

```
Res: 2.05 Content (application/json)
{
  "profile":[
    {
      "path":"1",
      "op":[3,4,7,11,12],
```

```
      "cf":[50],
      "m":[1]
    }
  ],
  "entity":[
    {
      "er":"coap://[sen5]/tmp,coap://[sen8]/tmp",
      "eo":["lst", "avg", "min", "max"]
    }
  ]
}
```

This simple example illustrates how an entity profile is constructed; either based on information from individual resource profiles or based on information retrieved via other means such as resources attributes derived from `/.well-known/core`. Much more information than shown here can be included and, by using a flexible representation format, the profile concept can be easily extended with new information.

The entity specific part of the profile currently supports the following fields:

- *Entity Resources (er)*: a list of the individual resources out of which the entity is composed.
- *Entity Message Type (emt)*: specifies the message type to be used for communication between EM and members. Possible values are *con* (confirmable) and *non* (non-confirmable). The default is *con*.
- *Entity Number of Replies (enr)*: specifies the number of replies that should be received from the members before sending a reply to the client. This makes it possible not to wait for all members to reply. The default behavior is to wait until all replies have been received or have timed out.
- *Entity Operation (eo)*: The operations that can be performed on the results obtained from the members. The operation is used to combine replies received from all the members (or the number of replies specified by enr) into one reply to the client. If at the time of querying the entity the client does not specify which operation to use, the first operation listed in this field will be used. Currently the following Entity Operations are supported:
  - *List (lst)*: A list of replies received from the members, without any arithmetic processing. This is the default behavior if no entity operation was specified.
  - *Average (avg)*: The average value.

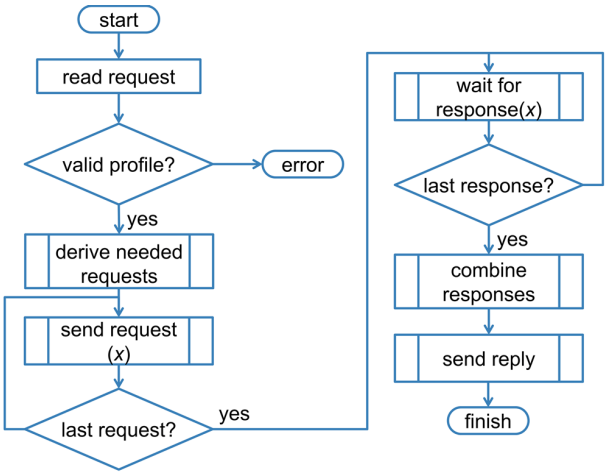


Figure 3.9: Simplified entity usage process flow.

  - *Minimum (min)*: The minimum value.
  - *Maximum (max)*: The maximum value.

- *Delay between Requests (delay)*: specifies the delay that should be injected between the requests sent from the EM to the members. The default is 0 and thus the EM will send the requests as fast as it can.

These entity profile fields can be provided by the client upon creation time. If no values are provided, the EM will use default values for the newly created entity. To construct the entity profile, the EM uses its internal knowledge to offer additional features that are not provided by the individual members. For example, the EM can interpret certain member payloads, convert between content formats and return the entity result in particular content format. Currently we support conversion between plain-text, JSON and RDFN3 content formats for numerical sensor values. The list of conversion functions can be extended easily.

3.5.4 Entity usage

Once an entity has been created, a response is sent back to the client. This response contains the URI of the entity, which was either requested by the client or assigned dynamically by the EM. The client can now interact with the entity by issuing a single CoAP request to the resource representing the entity. When a request for an entity arrives, the process flow shown in Figure 3.9 is executed. The EM breaks down the request into its components and sends the individual requests to the respective smart objects using unicast CoAP messages. It can either do that in

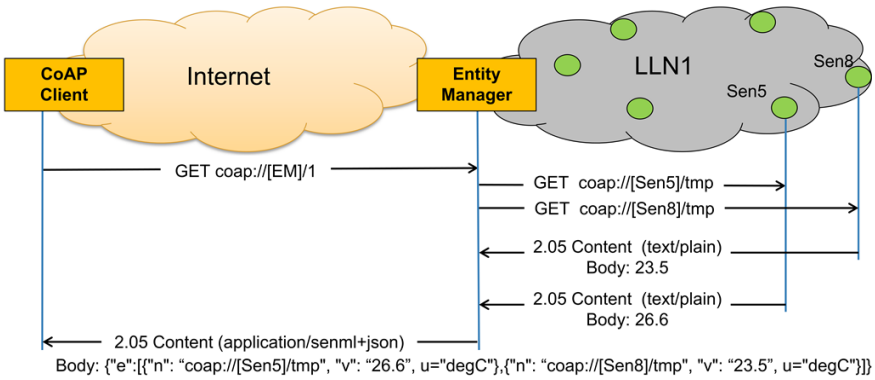


Figure 3.10: A CoAP client requesting from an EM to obtain the values for the entity that was previously created in Figure 3.7

parallel or sequentially with a configurable delay between requests to the members in order to avoid potential network congestion. Once all needed answers have been received, the EM creates a response to the client based on the individual responses and sends it to the client. Note that aspects such as how many members should respond, how the response is composed, how it should look like, *etc*. depend on the entity profile and can be customized using URI queries as will be explained later on.

Figure 3.10 shows an example of using the entity that was created previously in Figure 3.7. The client issues a GET request on the entity's resource `/1`. This results in the EM issuing two GET requests to the individual members, waiting for replies from both of them and then sending both results in one combined response back to the client.

The client can decide to query the entity using its default behavior as described in the entity profile or to customize its behavior. To customize the behavior the client can include URI queries in its request to the entity. The supported URI queries that can currently be used are: Entity Operation (eo), Entity Number of Replies (enr) and Delay Between Requests (delay) as described in Section 3.5.3. For example, to obtain the average value of the two temperatures of the entity `/1` in Figure 3.10, the client should use the URI: `coap://[EM]/1?eo="avg"` and should use `coap://[EM]/1?enr=1` to indicate to the EM that it is enough to send just any one of the two member replies as a reply to the entity. This last example demonstrates how our solution can be used to achieve a behavior similar to anycast requests when there are redundant members available.

3.5.5 Entity modification and behavior manipulation

It is possible that a client wants to modify an entity after its creation. For example, a client might want to add new members to the collection of members in the entity or remove a number of members. Alternatively, the client may want to customize the behavior of an existing entity. The latter can include aspects such as the default number or percentage of members that should respond before the EM replies to the client, the default content format of the response, the default operation (e.g., average, max, min, *etc*.) that should be performed on the results before sending them to the client, *etc*. Modifications to the entity or to its behavior can be made by updating the entity's profile and posting the updated information (PUT or POST) to `coap://[EM]/.well-known/profile?path=ENTITY_URI`, in which `/.well-known/profile` is a resource for accessing the profile of a resource as described in [25] and `ENTITY_URI` the URI of the entity, e.g., `/1` in our example. When a client wants to modify the profile of an entity, this information is passed to the EM, which will validate the request and change the profile if the validation was successful. Finally, removing an entity can be realized by sending a GET request to the entity management resource that includes `action=delete` URI query and specifying the entity to be deleted, e.g., `coap://[EM]/e?path=ENTITY_URI&action=delete`.

3.6 Implementation and evaluation

Our solution described above enables the use of unicast messages as an alternative to using multicasts for realizing CoAP group communication. In order to evaluate our solution and to show how it can be used in a real-world scenario we have implemented it and built a demo box for demonstration purposes. In this section we present the implementation of our solution and a basic description of the demo box followed by functional and performance evaluation.

3.6.1 Implementation

The key in our group communication solution is the EM. We have implemented the EM functionality on the gateway of the LLN using the CoAP++ framework [27]. The framework itself and the EM implementation on top of it have been realized in Click Router, a C++ based modular framework that can be used to realize any network packet processing functionality [28]. The CoAP++ implementation on the gateway also includes a resource directory and a cache that are used in the evaluation tests.

As group members we have used Zolertia Z1's boards [29] that run the popular Contiki 2.6 operating system [30]. This version of Contiki was the current version when we started our experiments and included a stable implementation of CoAP,

namely the Erbium CoAP server [31]. Our group communication approach does not require any changes on the CoAP enabled constrained devices. However, in order to demonstrate how the EM can use resource profiles to validate entities, we have added resource profiles to the constrained CoAP servers. Additionally, in order to be able to compare the performance of our solution with a multicast based solution, we have added multicast support to Contiki by using an open source implementation [32].

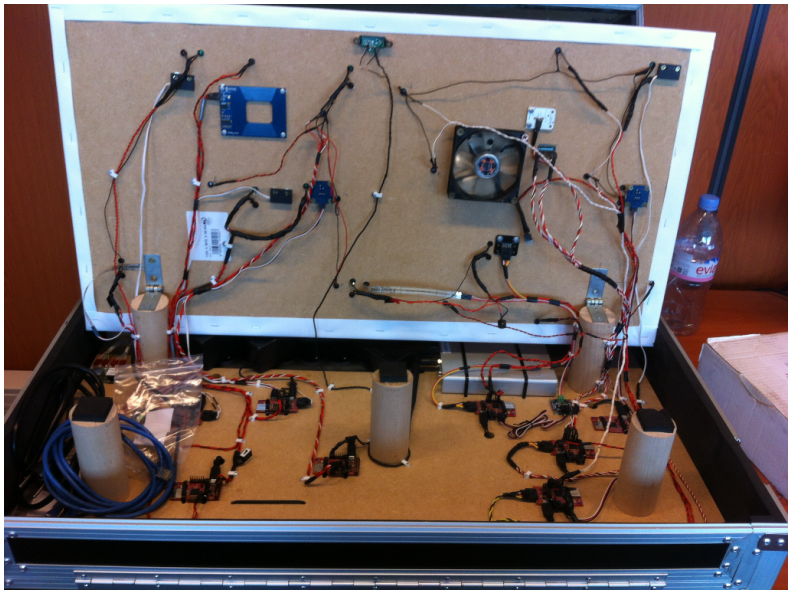
The CoAP++ framework's interoperability with the Erbium CoAP server as well with other CoAP implementations has been formally tested by the European Telecommunications Standards Institute (ETSI), a non-profit standards organization, in three events called CoAP Plugtests [33].

To demonstrate the practical use of our solution we have built a portable demo box (Figure 3.11). The box has two layers. The top layer has a floor plan of a house with several rooms. Each of the rooms is equipped with some wireless sensors and some wireless actuators. The top layer can be easily tilted to reveal the wireless sensor network that consists of eight wireless sensor nodes that are mounted on the back of the top layer and the bottom layer. The wireless sensor nodes are in the form of Zolertia Z1's boards that are running the Contiki operating system. Each of these nodes has been equipped with a number of sensors and actuators. The sensors include light intensity, temperature, proximity, movement (PIR), force, RFID and magnetic switch sensors. Supported actuators include multiple lights (in the form of LEDs) and a cooling fan. These sensors and actuators each have a corresponding CoAP resource. One of the wireless sensor nodes runs a 6LoWPAN border router and is connected to an Internet gateway. The gateway for this network is an Alix system board running voyage Linux. Apart from routing traffic, the gateway also provides the EM services.

Using this demo box we are able to show several home automation scenarios that use our group communication solution. For example it is possible to turn on all lights in a room (or a set of rooms) when the pressure sensor in the bed indicates that the person has left the bed while it is dark in the room. For more details about our demo box we refer to [34].

Besides its function as a showcase for our CoAP implementations, we have used the demo box for the functional evaluation of our group communication solution. However the demo box does not provide a suitable environment for good performance evaluation for several reasons. Since the box is relatively small, all radios are very near to each other and build a full-mesh single hop topology. This makes it impossible to perform multi-hop experiments. Furthermore, since the number of nodes in the demo box is only eight nodes, no larger scale experiments can be performed. Consequently, in order to obtain good insights in the proposed solution and to obtain a comprehensive performance evaluation it is needed to turn to either a simulation environment or larger scale testbeds. For this paper, we opted

(a) The upper level shows a map of a house with various sensors and actuators installed.



(b) Looking at the lower layer of the box the connections of the sensors are shown.

Figure 3.11: iMinds IoT portable home automation demo box.

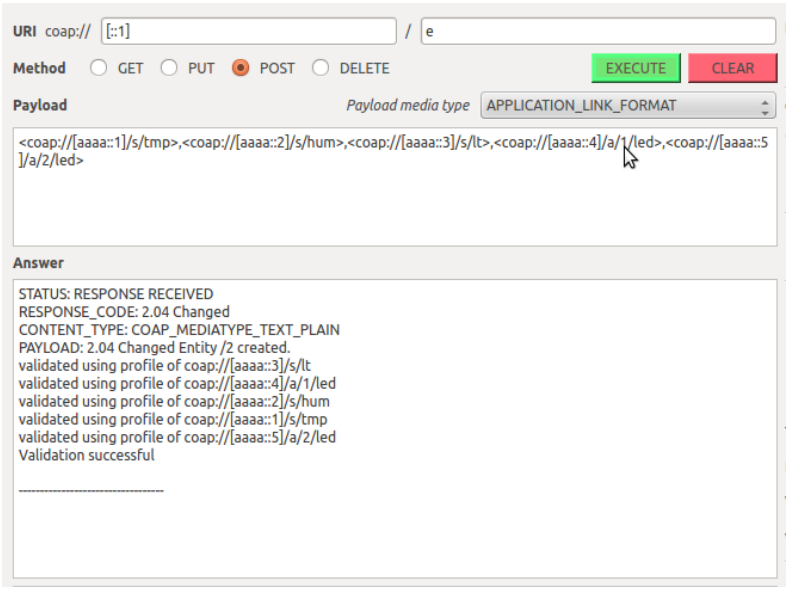


Figure 3.12: Sending a CoAP POST request to the EM to create an entity with five members, results in the creation of the entity with the URI /2 and in the validation of the entity.

for the first, *i.e.*, a simulation study using the Cooja network simulator, which is part of the Contiki operating system. The simulation environment allows both a functional and performance evaluation, with the demo box complementing the functional evaluation.

The simulation environment enables the initial evaluation of the performance of our solution for varying entity sizes and number of hops to the entity resources. Evaluation on larger real-life testbeds will prove useful for validating the simulation experiments and for conducting experiments in more dense and more realistic (e.g., Wi-Fi interference) environments. However this will take a significant amount of time and is beyond the scope of this work.

3.6.2 Functional evaluation

The functionality for creating, validating, using and deleting entities has been implemented as described above. In this subsection we demonstrate the main functionality of the implementation using a series of screenshots covering the life cycle of an entity. These screenshots are taken while communicating with the sensors in iMinds demo box. Figure 3.12 shows a screenshot demonstrating the result of sending a CoAP POST request to the EM to create an entity with five heterogeneous members. This request results in the creation of the entity with the URI `/2`

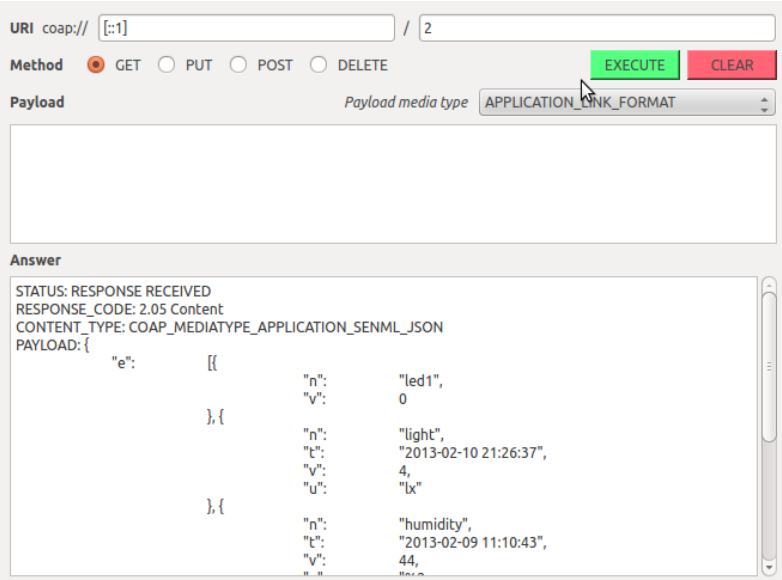


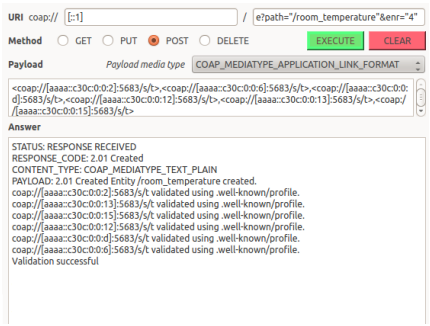
Figure 3.13: Sending a CoAP GET request to the entity results in a reply that combines the results of querying all members in the entity.

and in the validation of this entity by querying all members profiles. All complexity related to the creation and validation of the entity is hidden for the client and managed transparently by the EM. At this moment, the entity has been created and the client can use the newly created entity and interact with it by sending a single CoAP request to the entity resource.

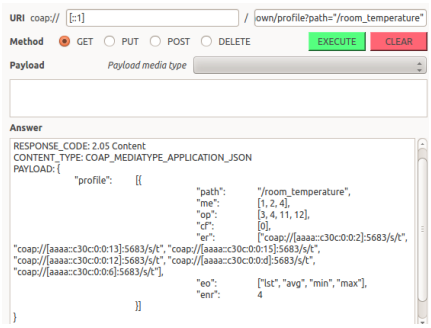
Figure 3.13 shows a client issuing a CoAP GET request to the newly created entity on the EM. The request ultimately results in a single reply from the EM, which combines the results of querying all five members of the entity. The client does not have to bother executing all individual requests and processing the corresponding results.

The above example demonstrates how an entity can be created and used with default values, since the client did not specify anything about its behavior neither at creation time, nor at usage time. However as described in Section 5, the EM allows the client to customize the behavior of the entity at creation as well as at usage time by using URI queries in the CoAP requests. Some of these features are shown in the example in Figure 3.14. In Figure 3.14a the client used URI queries at creation time to create an entity of six members, naming it `room\_temperature` and specifying that only four out of the six members need to reply before sending the combined reply to the client. Figure 3.14b shows the profile of the newly created entity, which, among others, shows that the entity supports four entity operations

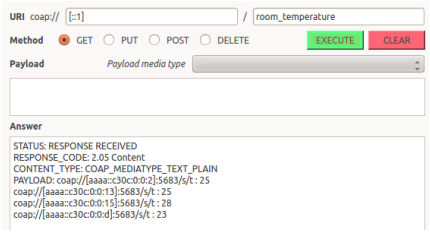
(eo=[`lst`, `avg`, `min`, `max`]) with `lst` being the default operation as it is the first operation listed. As expected, when querying the entity, the EM returns a list of the first four replies it has received from all members in a single JSON reply in Figure 3.14c. When the client uses the URI query (?eo="avg") to obtain the average instead of the default list, the individual responses are processed by the EM and the average value is returned as shown in Figure 3.14d.



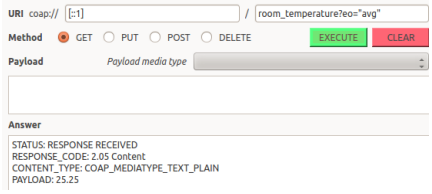
(a) Creation of an entity of 6 members and naming it "room_temperature" and specifying that only 4 out of the 6 members need to reply, before sending the combined reply to the client.



(b) The profile of the newly created entity.



(c) Querying the entity with default operation "lst".



(d) Querying the entity and specifying that the reply should only contain the average of the member values.

Figure 3.14: Advanced EM features.

In the last screenshot (Figure 3.15) we show how a client can select resources from a list of resources obtained from a resource directory to create an entity. The resource directory lists all CoAP resources of the sensors in our real-life wireless sensor network testbed, namely w-iLab.t [35]. Once the EM creates the entity, the resource directory is immediately informed about the newly created entity resource and uses this information to update the list of available resources.

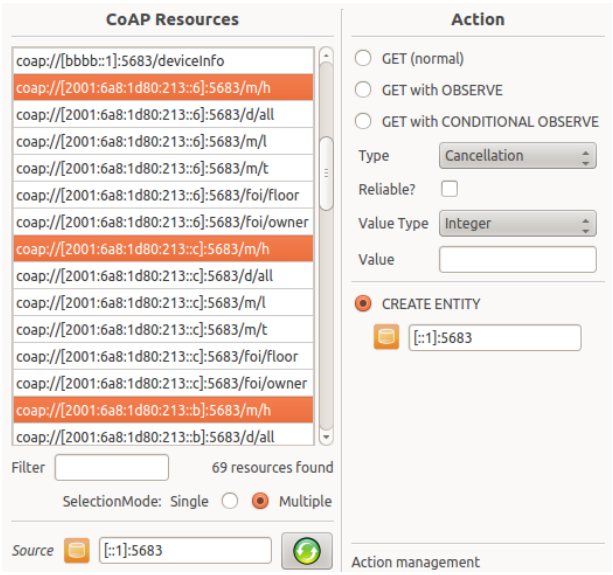


Figure 3.15: Creation of an entity by selecting three members from a list of resources provided by a resource directory.

3.6.3 Performance evaluation

In order to evaluate the performance of our group communication solution and compare it with multicast based solutions we performed a series of tests using the Cooja network simulator. In this subsection we present the results of these tests and analyze key performance indicators for both approaches.

3.6.3.1 Experiment setup

In our test we used a star topology with the gateway (node ID 0) in the middle of the five-leg star (Figure 3.16) and the nodes along the legs at 50 m distances. The transmission range (55 m) is enough to make the signal travel from one node to the other node on the same leg, but does not allow the signal to be heard between nodes on different legs. The interference range is 105 m, so that it covers the distance between three nodes on the same leg. The reason for selecting this topology is that it allows minimizing the impact of the underlying routing protocol on our measurements, as each node has only one deterministic route to the gateway. The gateway is running the example rpl-border-router provided by Contiki and therefor it is the RPL DODAG root, delegates the global IPv6 prefix and routes traffic to and from the constrained network. All other nodes run the Erbium server extended with resource profiles and multicast support as discussed in Section 3.6.1. Table 3.1 summarizes the parameters of the simulations in Cooja.

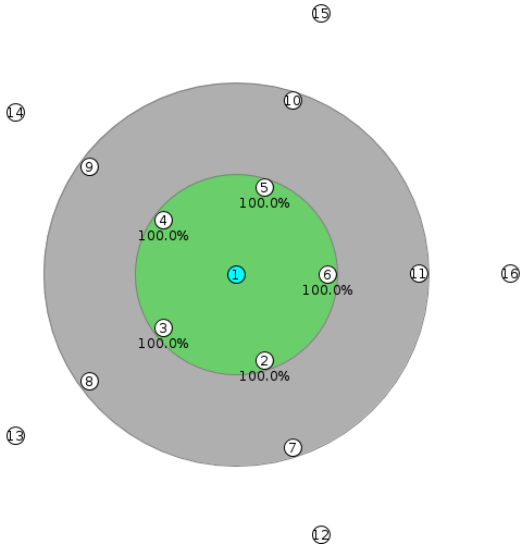


Figure 3.16: The network topology used in performance evaluation experiments. The inner circle shows the transmit range of the gateway and the outer circle shows its interference range.

Table 3.1: Cooja network simulator settings.

Radio Medium:	Unit Disk Graph Medium (UDGM): Distance Loss
Transmit Range:	55 m
Interference Range:	105 m
Distances between nodes:	50 m
Transmit Ratio:	100%
Receive Ratio:	50%–100% (depending on the experiment)

Since evaluating the performance of MAC and routing protocols is beyond the scope of this work, we had to take special care during the experiments to make sure that what we are observing is not a result of routing or MAC errors. For example, when introducing link errors in the simulations, routes can get lost or can be changed which has a considerable impact on the delivery rate of packets in the LLN. One solution would have been to use static routes. This is however not practical, since each node needs to be programmed with its own routes each time we change anything to the topology. To avoid such manual reconfiguration and since a stable version of RPL is available within Contiki, we used RPL as a routing protocol in all of the experiments. However, in order to minimize route changes from impacting our results we have taken the following two measures in all experiments:

1. Before sending any CoAP messages in the LLN, we wait for some time to allow RPL to establish routes to all nodes. In our chosen topologies, waiting for 2 min was in most cases sufficient to achieve this goal. In a few cases (with a high percentage of packet loss) we had to wait more than an hour, since the time between RPL neighbor updates is exponential up to a certain value.

2. The Contiki implementation of RPL relies on Contiki to maintain the neighbor table. Contiki in turn removes a neighbor if it does not hear its heartbeat a consecutive number of times (three by default). In very lossy networks (such as in our experiments with high packet loss) this behavior might lead to neighbors being removed and thus all routes via that neighbor as well. However in all of our experiments the nodes are static and never disappear from the network. As such they should not be removed from the neighbor table. To achieve this goal we have changed the Contiki configuration parameter UIP_CONF_ND6_MAX_UNICAST_SOLICIT from its default value of 3 to 100. This allowed all experiments to be completed without routes to the nodes being lost.

With these two measures in place it was possible to get stable routes during the experiments. The other factor that may heavily impact the measurements is the used MAC protocol. In order to save energy, MAC protocols for LLNs use Radio Duty Cycle (RDC) to shut down their radios when not in use. Contiki has its own MAC protocol, called ContikiMAC, with a good RDC schema. However ContikiMAC requires more resources than other MAC protocols. In our experiments we also need, next to CoAP and multicast, debug information in order to be able to collect measurement statistics. As such, it was impossible to fit CoAP, multicast, RPL, debug info and ContikiMAC on the used Z1 motes. Instead of ContikiMAC we therefore used null-rdc, which is Carrier Sense multiple Access (CSMA) MAC protocol without RDC (radio always on). While using null-rdc is not realistic for battery-powered devices, it still helps avoiding delays in the collection of measurement statistics as imposed by the RDC protocols. However, to still get an idea about the impact of RDC on our solution, we repeated some of the tests using Xmac. Xmac uses a simple RDC, but has less stringent requirements in terms of memory consumption than ContikiMAC.

In all the experiments presented in this subsection the multicasts were sent using non-confirmable CoAP messages as required by the group communication draft [14] and the unicast were sent using confirmable CoAP messages to achieve reliability.

3.6.3.2 Congestion control optimizations

An important aspect of group communication is congestion control, especially in LLN where resources are limited. Network congestion can lead to extended response times and significant energy consumption, due to frequent retransmissions of packets. CoAP provides basic congestion control by using the exponential back-off mechanism (Section 3.2.1) and by limiting the number of open requests from a client to any server to one request by default. Furthermore, CoAP specifies that, when using multicasts, a certain random delay should be inserted before replying to multicast requests. In CoAP terms, this delay is called Leisure. The server could either use a default value for Leisure or compute a value for it. If the server has a group size estimate G, a target data transfer rate R and an estimated response size S, a rough lower bound for Leisure can then be computed as:

$$Leisure_{lowerbound} = \frac{S * G}{R} \tag{3.2}$$

When only taking into account the 1-hop neighbours of the gateway in our test network in Figure 3.16 G equals 5, S equals approximately 80 bytes, and the target rate can be set to a conservative 8 kbit/s = 1 kB/s. The resulting lower bound for the Leisure is then equal to 0.4 s. However, since CoAP servers will often not be able to compute the Leisure, we elected to use the default Leisure value (5s) in all of our multicast experiments. For a more complete discussion of the Leisure period and its estimation we refer to Section 8.2 of [5].

CoAP does not specify a congestion control mechanism when a single client is communicating with many servers using unicasts as is the case in our group communication solution. However our experience shows that this can quickly lead to congestion. A simple solution for avoiding network congestion when using unicasts is to limit the rate at which requests are sent. In order to examine this, we conducted a series of experiments to query an entity of five members and measure the response time, which is expressed as the time between the moments the client issues the request to the EM until it gets back the response. We repeated the same experiment for different delays between the requests sent from the EM to the members. We repeated the experiment 50 times for each setting and computed the averages. The same set of experiments was repeated when all members were either one or two hops away and while using null-rdc or Xmac.

As expected, the experiments revealed that the average response time of an entity can be improved by inserting small delays between the requests for individual entity members (Figure 3.17). However, the best delay depends on both the topology and the used MAC protocol. This is most obvious in the graph of Xmac for 1-hop communication. Here one observes two peaks at about 100 ms and 400 ms. The first peak is a result of collisions between the forwarded requests from the nodes at the first hop to the nodes at the second hop with the delayed

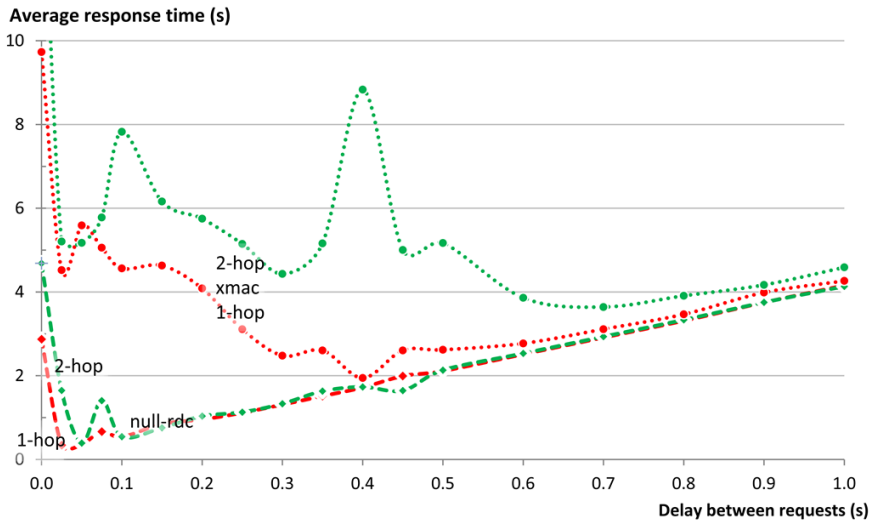


Figure 3.17: The average entity response time can be improved by inserting small delays between the requests for individual entity members. The best delay depends on the topology and the used MAC protocol.

requests for the next member. The second peak is a result of collisions between the replies from the members at the second hop with the delayed requests for the next member. For delays less than 0.5 s the used MAC protocol had more effect on the response time than the delay inserted between the requests. However as the delay between requests grows larger it becomes the dominating factor for the total response time with a linear relationship between the two.

In the remaining experiments we used only null-rdc as MAC protocol for two reasons. First, we wanted to avoid measuring the delays in communication as imposed by the use of Xmac. Second, Xmac does not support multicast well, and thus we would not have been able to make a fair comparison between our group communication solution and multicast based solutions.

For the star topology which we used in most of our experiments, a delay of about 50 ms provided the best response when null-rdc was used. As a result we have used a 50 ms delay between requests to the members in all other experiments discussed later in this section. In addition to this delay there are other delays that impact the communication. The nodes and the EM need some time to process the CoAP packets they send and receive (e.g., time needed by nodes to prepare the CoAP reply and the delay needed by the EM to combine all replies into one reply to the client). We call the sum of all such delays the Processing Delay D_p. Finally we call the sum of the average signal propagation time and the average time need to send and receive a packet over any transmission link the Transmission delay

D_t. We have experimentally evaluated D_p and D_t by averaging the values for 100 transmissions for our network topology when communicating with the nodes that are only 1-hop, 2-hops or 3-hops away. Table 3.2 provides a summary of the delays in our experimental topology.

Table 3.2: Summary of communication delays in used topology.

Entity Delay between Requests to Group Members:	$D_e = 50ms$
Processing delay:	$D_p = 120ms$
Transmission delay:	$D_t = 23ms$

3.6.3.3 Reliability

Reliability is a key performance indicator. In this subsection we first present the theoretical model for calculating the reliability of the two group communication approaches and then present the results obtained from our simulations.

Theoretical Calculation

Let us assume that the probability of losing a packet when it is sent over any link in a lossy network is equal to l. Thus the probability for success at any link-transmission in the network equals $1-l$. If the communication is over N-hops, the number of links equals $2N$ and the probability that the communication succeeds over the link-transmissions (since every link has to be crossed once for the request and once for the reply) is:

$$P_{N_{non}} = (1-l)^{2N} \tag{3.3}$$

This communication success probability applies for multicast communication as well as for non-confirmable (non) CoAP unicast communication. However, when using confirmable (con) CoAP unicast communication as the case in our group communication solution, CoAP tries to achieve reliability by using a simple stop-and-wait retransmission with exponential back-off (see Section 3.2.1). This means that if a reply to a confirmable packet is not received within the back-off time, the CoAP sender will retransmit the packet. If a reply to the first transmission is not received, CoAP will retry the transmission until MAX_RETRANSMIT (by default 4) is reached. Again these retransmissions have the same probability for success as the first attempt. And thus the probability that retransmissions are needed for successful transmission over N hops (r can go from 0 to 4) is:

$$R_{N_r} = P_{N_{non}}(1 - P_{N_{non}})^r \tag{3.4}$$

Thus the probability for success when using CoAP con communication equals the sum of the probabilities of successful communication of any of the five transmission attempts:

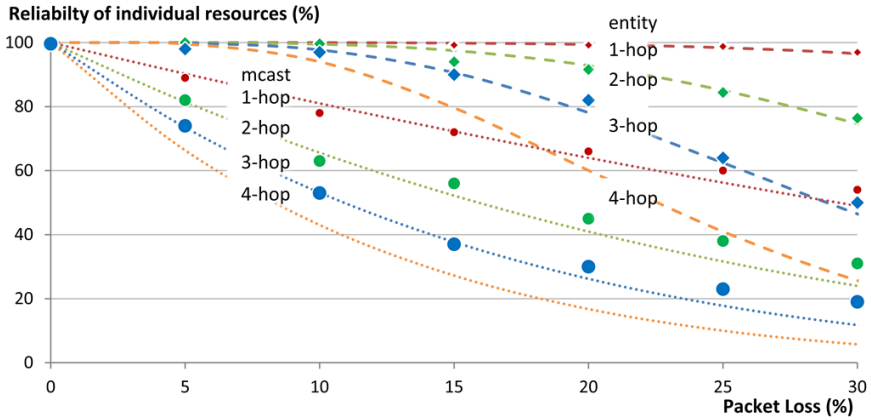


Figure 3.18: The reliability of individual group members is a lot better when using entity based group communication.

$$P_{N_{con}} = \sum_{r=0}^{4} R_{N_i} = 1 - (1 - P_{N_{non}})^5 \tag{3.5}$$

In many group communication use cases, it is desirable to get answers from all members of the group. A complete group communication is considered successful when all members in the group also have successful communication:

$$\begin{aligned} P_{NG_{non}} &= (P_{N_{non}})^G \\ P_{NG_{con}} &= (P_{N_{con}})^G \end{aligned} \tag{3.6}$$

This reliability of the unicast group communication is achieved by relying only on default CoAP retransmissions. If higher reliability is desired, the EM can perform its own retransmissions or fine-tune the default CoAP retransmission settings on an entity-wide level or per member.

Experimental Evaluation

To measure the reliability we used the same star topology to communicate with a group of five members that were either 1-, 2-, or 3-hops away from the gateway. We varied the percentage of packet loss in the network in 5% steps and measured the reliability of getting responses to the respective requests. We repeated the same experiment for our group communication solution and for multicasts. We run each experiment 50 times. Figure 3.18 shows the effect of packet loss on the communication reliability in our 1-, 2-, and 3-hop star network. Multicasts are not transported reliably and thus reliability of the network decreases as soon as there

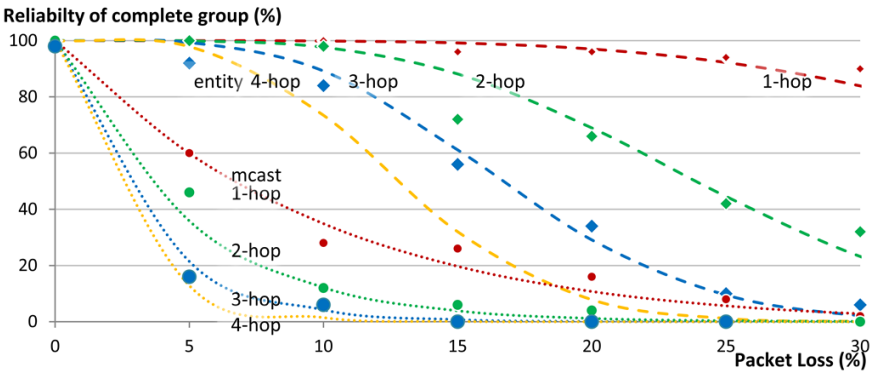


Figure 3.19: The reliability of the complete group is less than the reliability of individual members (Figure 3.18). Again, the reliability of the complete group is a lot better when using entity based group communication.

is packet loss in the network. When using our unicast group communication solution, CoAP confirmable messages are used. In our 1-hop test topology reliability was higher than 99% even when the packet loss of the network reached 25%. At 30% packet loss the reliability is reduced to 97% (compared to 49% in the case of multicasts). Figure 3.18 also shows that the packet loss increases with an increasing hop count, both for unicast and multicast communication. This is due to the fact that every message (both request and reply) between a client and a server has an additional chance of getting dropped at each hop on the way to its destination. Nevertheless, in our 2-hop network 100% reliability was maintained for unicast communication until a packet loss ratio of 10%. In the 3-hop network the unicast reliability started dropping below 100% already by 5% packet loss. The dashed and the dotted lines in Figure 3.18 are the theoretically expected values up to networks with 4-hops, while the points are the actual obtained results from the experiments (up to networks with 3-hops). It is clear that there is a good match between both.

Figure 3.19 shows the effect of packet loss on the reliability of the complete group in our 1-, 2-, and 3-hop star network. Certainly the reliability of a complete group is less than the reliability of its individual members, since the loss of a message to or from a single member, renders the complete group request unsuccessful. In these cases the use of multicasts does not provide good results. Already at 5% packet loss the reliability of a 1-hop network drops to 80% and below 20% for a 3-hop network. In contrast, our unicast based group communication maintains 100% reliability in the 1-hop network and only drops to 92% in the case of 3-hop network. Figure 3.18 shows good match between the theoretically expected values (the lines) and the actual obtained results from the experiments (the points).

3.6.3.4 Number of packets

Another key performance indicator is the amount of energy consumed by the network to complete the communication task. The main two contributing factors to the energy consumption are the efficiency of the RDC and the numbers of packets sent. The RDC mostly depends on the MAC protocol and is beyond the scope of this article. We use the number of packets sent inside the LLN as an indicator for the amount of energy consumed by the network as exact numbers for energy consumption are hardware dependent. In this subsection we first present the theoretical model for calculating the number of packets and then present the results obtained from our simulations.

Theoretical Calculation

When using multicast group communication, the number of packets that are sent depends on the topology of the network. In our star topology, all nodes that are one hop away can be reached with just one multicast packet. After the first hop, the paths are not shared by any node and the packets have to travel similar to unicasts along them. When the requests reach their destinations, replies are generated and are sent back using non-confirmable unicast messages. Thus, in the case of a successful communication, the number of transmitted packets can be obtained as follows:

$$pkts_{N_{sucess}} = 1 + G(2N - 1) \tag{3.7}$$

Taking into account the probabilities of failure at each link-transmission (see Section 3.6.3.3), the average number of transmitted packets when using multicast group communication can be obtained as:

$$pkts_{N_{mcast}} = \left(\sum_{i=0}^{2N-1} l(1-l)^i \times (iG+1) \right) + (1-l)^{2N-1} \times (1 + G(2N-1)) \tag{3.8}$$

This formula takes into account the fact that the request or reply can get lost at any intermediate hop (total of $i = 0, ..., 2N - 1$ possibilities with the probability $l(1 - l)^i$), resulting in a different number of packets being transmitted $(iG + 1)$.

When using our group communication solution, all transmissions are unicast and thus in the case of a successful communication, the number of transmitted packets can be obtained as follows:

$$pkts_{N_{sucess}} = 2NG \tag{3.9}$$

And the average number of packets in case of failure (*i.e.*, packet loss somewhere on the path) as:

$$pkts_{N_{failure}} = \begin{cases} \frac{G\sum_{i=0}^{2N-1} l(1-l)^i \times (i+1)}{\sum_{i=0}^{2N-1} l(1-l)^i}, & l > 0 \\ 0, & l \leq 0 \end{cases} \tag{3.10}$$

The formula is normalized as the sum of all probabilities in the nominator is not 1 since it does not include the probability of success over all links P_N. To avoid division by zero we set $pkts_{N_{failure}} = 0$ when $l \leq 0$.

And the average number of transmitted packets in these cases of retransmissions can be calculated as:

$$pkts_{N_r} = r \times pkts_{N_{failure}} + pkts_{N_{sucess}} \tag{3.11}$$

If the last retransmission attempt fails, the sender is notified about the failed transmission. Thus the probability for failure for an N-hop communication:

$$R_{N_{failure}} = (1 - P_N)^5 \tag{3.12}$$

And the average number of transmitted packets in this case can be calculated as:

$$pkts_{N_5} = 5 \times pkts_{N_{failure}} \tag{3.13}$$

Thus the average number of transmitted packets for N-hop confirmable CoAP unicast communication can be calculated as follows:

$$pkts_{N_{entity}} = \left[\sum_{r=0}^{4} R_{N_i} \times pkts_{N_r}\right] + R_{N_{failure}} \times pkts_{N_5} \tag{3.14}$$

Experimental Evaluation

When multicasts are used, one request is sent to multiple destinations, and one reply is sent by each member. For our five nodes that are 1-hop away, assuming there is no packet loss in the network, the number of transmitted packets is thus six packets (an average of 1.2 packets/member). When the network is lossy the number of packets can only become less, as packets are being dropped. This can be clearly seen in Figure 3.20 for 1-, 2- and 3-hop networks. On the other hand, when using unicasts the number of packets increases as the loss increases. This is a logical result of packets being retransmitted by CoAP to compensate for the packet loss. The dashed and dotted lines in Figure 3.20 are the theoretically expected values, while the points are the actual obtained results from the experiments. It is clear that there is a good match between both.

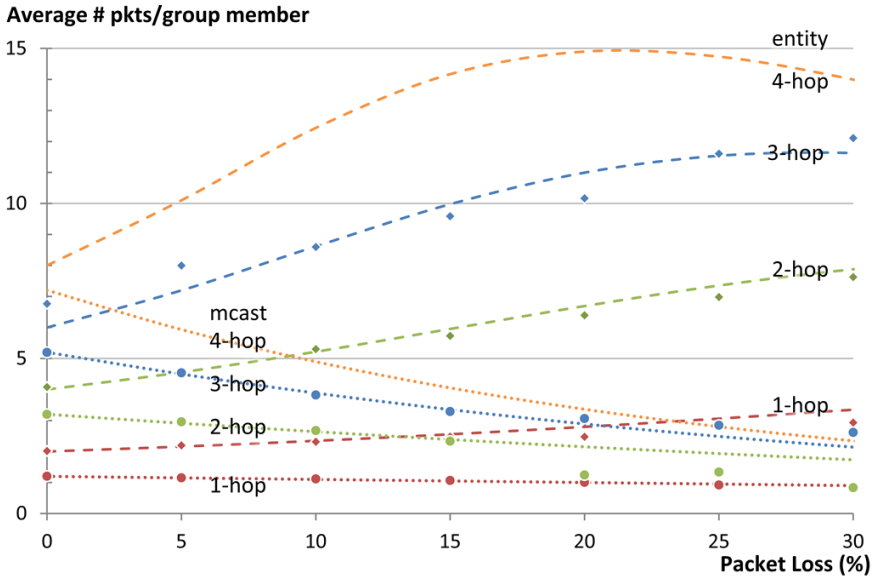


Figure 3.20: The number of packets sent in the LLN for each group member at different packet loss levels. For multicasts the number of packets decreases as the loss increases. For unicasts the number increases due to CoAP retransmissions.

3.6.3.5 Response time

Another key indicator of the performance of any group communication solution is its response time. Figure 3.21 shows that the average response time when using our group communication solution increases with the increase of packet loss in the network. When there is no loss, both group communication methods have similar response times (0.7 s for multicasts and 0.8 s for our solution). When the packet loss increases, the response times for multicasts remain constant, since either the packet is delivered on time or it is just dropped. In the case of our solution, when a packet is dropped CoAP attempts to retransmit it, leading to an increased overall response time. At 15% packet loss, the average response time for the 1-hop network is about 3.5 s and increases to about 10 s at 30% packet loss.

The CoAP retransmission behavior can be clearly seen in Figure 3.22, which shows the same results of Figure 3.21 in the form of a scattergram. It shows how the response times of the unicasts are grouped along the time axis. Replies with a response time of around 1 s were sent without any retransmissions. Replies with a response time around 7, 15 and 30 s occur when one, two or three retransmissions take place respectively. The last group between 60 and 100 s is for replies that were received after four retransmissions, or those that timed out without a reply.

An interesting aspect to consider is the impact of long delays in network on

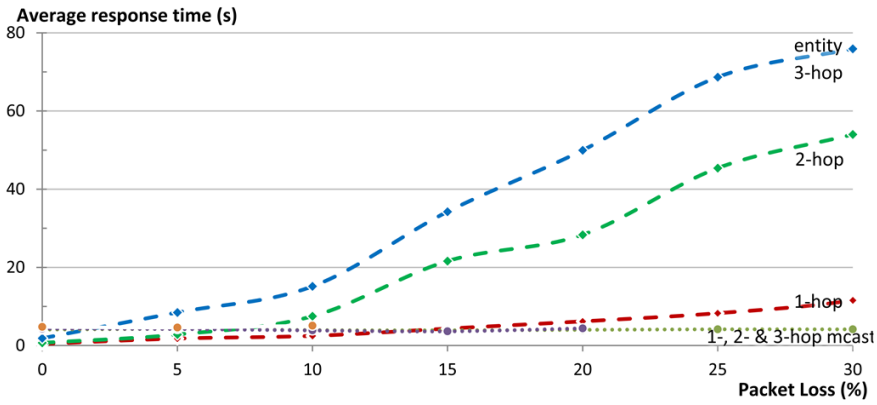


Figure 3.21: Average group response time.

the client. In our case both the communication between the client and EM from one side and the communication between the EM and the members on the other side are using the same CoAP time out mechanism with default values. Thus it is expected that the client might time out and start retransmitting the request before the EM requests to the members are answered or have timed out. To avoid these unnecessary retransmissions from the client, the EM responds to the client with an empty acknowledgment message as soon as it receives the request, indicating that the EM is processing the request and will send a separate reply once an answer is ready. As a result the client does not send any further retransmissions of the request.

3.6.3.6 Impact of caching

As mentioned in Section 3.2.1 CoAP foresees a freshness model for responses. A CoAP server may include a Max-Age option in the reply to indicate the maximum time a response may be cached before it is considered not fresh. If this option is not included in any CoAP response, it can be assumed that the response will be fresh for 60 s and thus will not be queried again by a cache within this time frame. Max-Age values in responses should take into consideration the frequency at which the value of the corresponding resource changes. By doing so, unnecessary queries to constrained devices can be heavily reduced, especially for resources that do not change frequently.

When a client makes a multicast request, the cache always makes a new request to the multicast group (since there may be new group members that joined meanwhile or ones that did not get the previous request) (see Section 8.2.1 of the CoAP draft [5]). So, the main problem is that the cache is not aware of all receivers in a multicast group. This information is needed in order to be able to verify whether

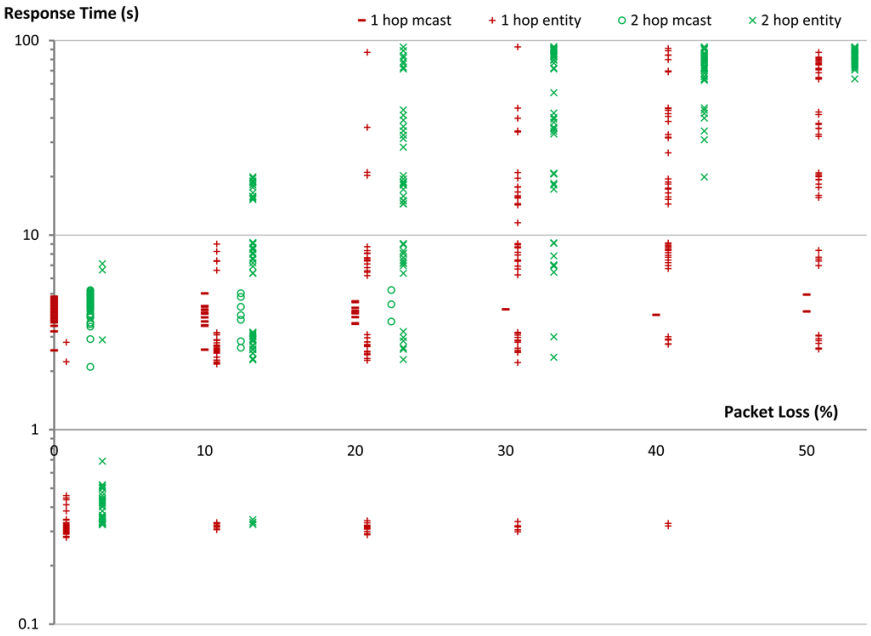


Figure 3.22: Scattergram of the response times for 1- and 2-hop networks using multicast and unicast group communication for different packet loss values.

a response for all receivers in the multicast group has been cached. If the cache knows this information (which is not the case for a normal cache), the cache could serve the multicast request. But even then the multicast has to propagate in the network as soon as one of the responses is missing.

In order to test the behavior of both group communication approaches when caches are used, we have conducted a series of tests using our star topology in its 1-hop configuration with five members. In the tests we varied the time between requests to the group in 5 s increments and measured the number of packets transmitted inside the LLN and the response time. We sent 60 requests per experiment. The experiments were conducted without introduction of any packet loss. The average number of packets transmitted inside the LLN is shown in Figure 3.23. As expected when using multicasts, regardless of the frequency at which requests are sent, there was always one request and five replies from the five members resulting in an average value of 1.2, 5.2 and 7.2 packets/member for group members that are 1-, 2- and 3-hops away from the EM respectively. On the other hand, when using our unicast based solution the average number starts at 0.04, 0.07 and 0.1 packets/member and increases in steps to reach two, four and six packets/member for group members that are 1-, 2- and 3-hops away from the EM respectively. The reason for this behavior is that the initial requests are always sent to all members as

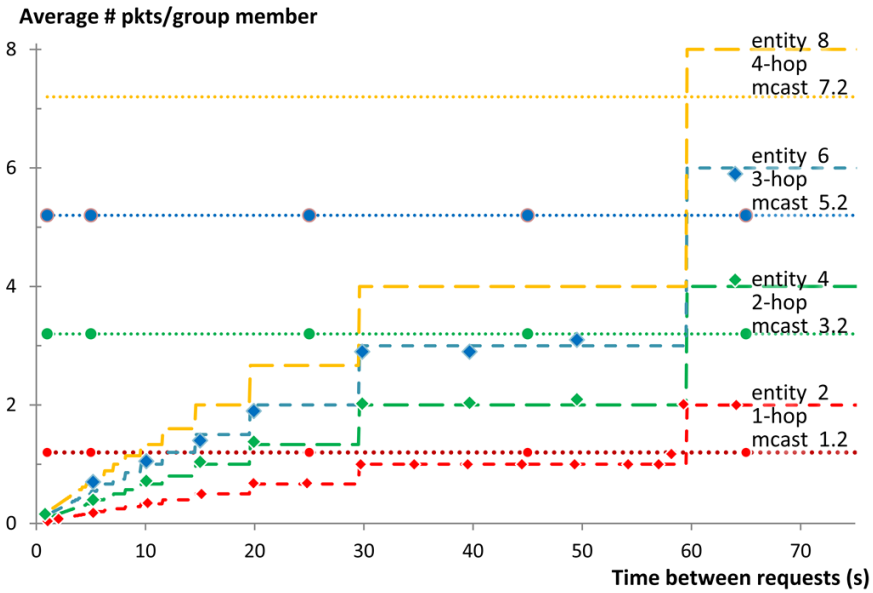


Figure 3.23: When requests are served from the cache the number of packets transmitted inside the LLN is reduced.

the cache is still empty. Since the replies in our example did not include the Max-Age Option, the cache assumed that the replies were valid for the default value of 60s. Subsequent requests were thus served from the cache as long as the first replies were still considered fresh. Once the Max-Age value expired new requests were sent to the members again.

Figure 3.24 shows the response times for the requests for both group communication approaches, when a cache is used for the cases in which the group members were 1- and 2-hops away from the EM. For multicast the response time is between 2 and 6 s regardless of the delay between requests. In the case of our group communication approach one can see that there are two groups of replies. The first group is between 0.20 and 0.25 s and the second group is higher than 0.3 s. The first group represents the replies that were served from the cache, while the second group represents the replies that were not fresh in the cache and thus obtained from the group members. With the increase of the time between the requests, one can observe how the number of replies served from the cache decreases. When the time between requests becomes larger than Max-Age, the number of requests served from cache becomes zero.

The cache stores the individual replies from the group members. This makes it possible to benefit from the cache even when different clients request a different Entity Operation. For example if two clients access the same entity within Max-

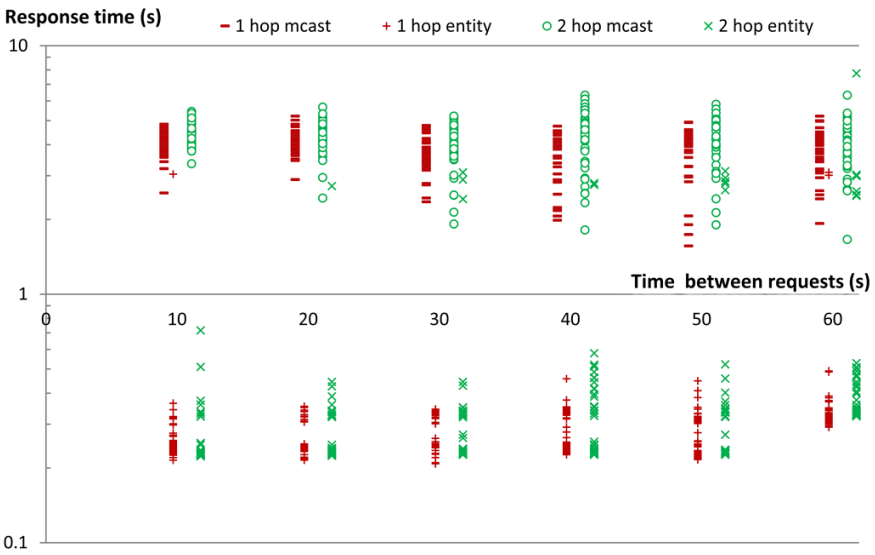


Figure 3.24: Responses that are served from the cache are much faster than responses obtained from the nodes inside the LLN.

Age, but one is requesting the average and the other is requesting the maximum value, then each entity member will be queried only once. The EM then processes the replies from the members and the cache and assembles the reply to the client as was requested. The same applies if a single member is part of two different entities (overlapping entities). The member will be queried only once within Max-Age, regardless of the entity that is requesting it. Finally, the cache can be populated via other requests/responses that pass via the gateway; it's not limited to just the requests/responses employed by the EM.

Please note that caching is very interesting, but it will not work for actuators. However, for actuators one probably wants high reliability, which is again in favor of our solution.

3.6.3.7 Entity validation overhead

As described in Section 3.5.3, the EM can perform a validation process on the entity at creation time. In this subsection we measure the communication overhead of this validation process as a function of the number of group members. For this, we need a topology in which all group members are located at the same distance (in hops) form the EM. Otherwise the results will be influenced not only by the group size, but also by the hop count. Therefore, we use a single hop topology consisting of several nodes in a single collision domain. Every node in this LLN had exactly one resource that is part of the entity. This topology minimizes the

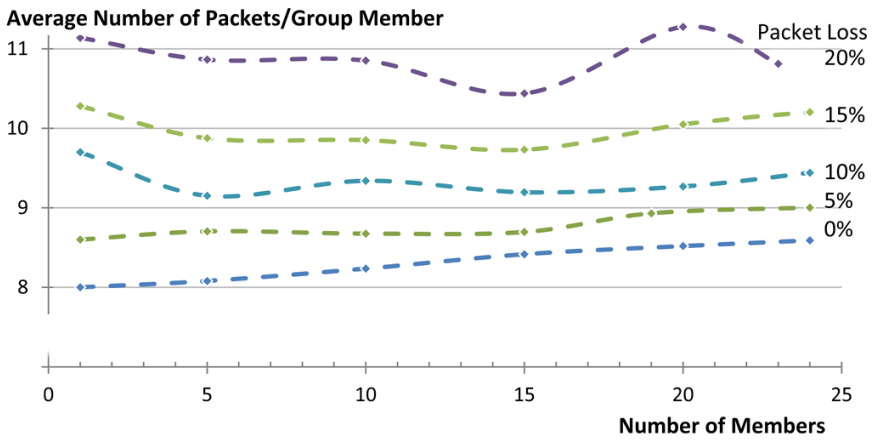


Figure 3.25: The number of required packets needed to validate a single entity member increases slightly (due to CoAP retransmissions) with the increase of packet loss in the LLN. As a minimum eight packets are required to obtain the profile of the respective member.

effect of routing and forwarding delays on the validation process. For this topology, we performed several experiments and measured the number of packets sent inside the LLN during the validation process and the time that was needed to complete the process. We did not utilize the cache in this experiment. The experiment was conducted for several entity sizes; starting with a single member entity up to an entity with 24 members. The experiment was repeated for several packet loss percentages between 0% and 20%.

In order to validate an entity member, the EM starts by querying the profile of that resource. In our case the profile was 107 bytes long. A minimum of eight packets was required to obtain it (Figure 3.25). The reason for this is that the maximum usable payload size in our LLN is about 60 bytes (see Section 3.2.3). This means that blockwise transfer should be used to obtain the profile. As mentioned in Section 3.2.3, CoAP utilizes predefined block sizes. Unfortunately we cannot use the block size of 64 (since it is just above the usable payload size) and have to use 32-Byte blocks. Consequently, four packets are needed to obtain the complete resource profile and another four packets to acknowledge the receipt of the requests and transfer the profile parts. Figure 3.25 shows that in the simplest case (one member in the entity and no packet loss) exactly eight packets were needed. When the number of members in the entity increases, so does the average number of required packets. This increase is due to the fact that the probability of packet drops due to collisions increases with the increase of the collision domain size. Since we are using confirmable CoAP messages, CoAP utilizes its retransmission mechanism to obtain the missing packets. Similarly, if we inject random packet

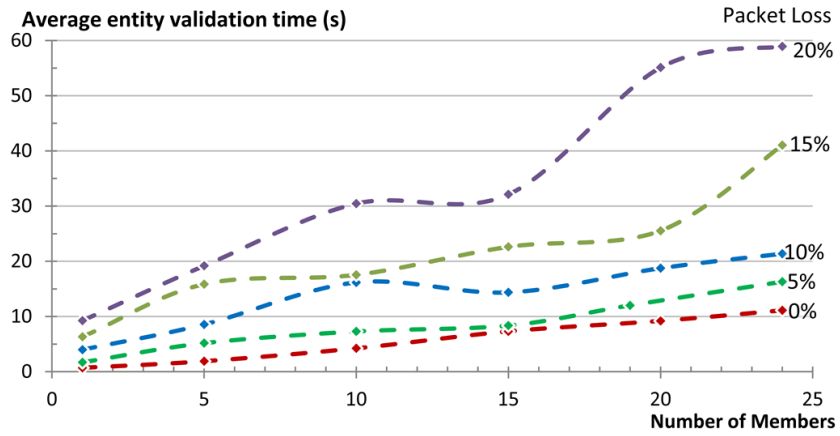


Figure 3.26: Average entity validation time.

loss in the network more packets will be transmitted to compensate for the loss. In addition to the CoAP retransmissions the EM uses its fallback mechanism for validation. This means that for cases with a lot of packet loss, it may happen that all retransmission attempts to obtain the profile fail. In these cases the EM tries to check if the resource is available based on the content of /.well-known/core of the node. Here again, blockwise transfer is used to obtain the information in 16 packets (eight requests, eight replies). If this also fails, the EM tries to query the resource to check if it exists, consuming two packets (one request, one reply).

As expected, Figure 3.26 shows that the average entity validation time increases when the number of members in the entity is increased since validation requests are sent sequentially. The validation time also increases with the increase of packet loss in the LLN. The increase here has more profound effect on the delay since CoAP doubles the timeout between retransmissions, explaining the exponential trend in the delay.

Certainly, the entity validation overhead has negative impact on the energy consumption of the nodes in the LLN. If an entity will be created and used for a very limited time, this impact cannot be ignored. However, in many use cases it is expected that, once created and validated, entities will have a long life span. For example an entity that represents the lights in a certain room will change only if the layout of the room and lights change, which is in many cases a relatively rare occasion. In these cases the communication overhead of validation can be ignored. Additionally, one should also consider that the information needed for validation (profile; /.well-known/core) is fairly static and thus can be efficiently cached by having a very-long max-age value. In such cases the validation can be based on the information obtained from the cache and thus be done almost immediately, without the need for packet transmissions inside the LLN.

3.7 Discussion

In this section we look again at the requirements for the IoT scenarios as presented in Section 3.3 and discuss how our solution addresses these requirements. We also provide the drawbacks of our solution in light of these requirements and the insights gained from experimental evaluation of our solution.

1. *Flexibility*: our CoAP group communication solution is highly flexible when compared to other solutions. Other than supporting the standard base CoAP protocol, group members do not have to support any additional extensions. Since group definition and behavior are set on the EM only, changes and extensions can be done on the EM without the need for any changes on the individual members. We support homogeneous and heterogeneous entities in terms of their CoAP resources. A group can be built from members as along as they have a shared subset of CoAP methods. Members do not have to offer the exact resource name or the same content-type, since the EM can take care of conversion when possible. Since we use IPv6 unicasts for communication between the EM and the members, the members can be located anywhere on the IPv6 Internet as long as they are reachable.

2. *Light-Weight (footprint)*: since our solution does not require any additions to standard CoAP implementation, it can run on Class 1 devices as already demonstrated in the three CoAP plugtest events. Furthermore the members do not store any information about which groups they are part of. Thus our solution scales well with the number of groups a certain member can be part of.

3. *Use of Standards*: our Solution is based on CoAP, which is expected to become the application layer standard the IoT. We rely only on the base CoAP standard and require no optional extensions to it. However, if available, the resource profile extension can be used to better understand the capabilities on the individual members in order to apply advanced features such as content type conversions.

4. *Performance*: as our solution uses only standard CoAP over unicast UDP/IPv6 it is a cache friendly solution. Caches are a powerful tool in reducing the number of messages inside LLNs and enhancing response times. However only CoAP GET method can be cached and thus no benefit of caches can be expected for the other CoAP methods (*i.e.*, PUT for actuator data). Nevertheless, in these cases it is often more important to reliably deliver the message even if performance is reduced as a result.

5. *Validation and Error Handling*: we support heterogeneous members that might have properties that cannot be combined. Thus whenever a group is

created the EM tries to validate the group in order to make sure that the group works as intended and informs the creator about the validation results. By giving the possibility to build the reply from just a subset of the members, the group becomes tolerant in handling certain error conditions such as node or route failures.

6. *Reliability*: by using CoAP confirmable messages we rely on CoAP retransmission mechanisms to handle reliability of communication with the entity members. In addition it is possible for the EM to customize the parameters for retransmissions (number of retransmissions and time-outs) for all or certain group members based on the history of communication with these members.

7. *Ease of Group Manipulation*: if the needs of the user change with time, our solution enables changing group memberships based on these needs by just changing the entity definition on the EM. The group members are not involved in this change and do not need to be reprogrammed or reconfigured in any way. By integrating the resources of the members and the EM into a resource directory, it becomes easy to locate and combine resource into new entities. This can benefit both M2M resource discovery as well as end user using Graphical User Interfaces (GUIs) to browse and build new entities.

8. *Expressiveness/Control*: by supporting processing of individual results at the EM, we enable greater control over the results that are sent to the user as a result of contacting the group members. Currently we support sending all replies as a list, or combining the results in one arithmetic value (minimum, maximum, or average). We also support the user to specify the number of members that should reply before an answer is composed and sent back to the requester. Additional features to improve the control over entity behavior can be easily added by extending the EM functionality without the need to extend the members themselves.

9. *Security*: by using only standard CoAP unicast messages for communication, we make it possible to apply whatever used transport security mechanisms (such as DTLS) to these messages as well.

It is clear from the discussion above, that our solution is capable of addressing the requirements we have put forward for group communication in IoT scenarios. Of course, the gain in flexibility comes with some drawbacks as well. Compared to a multicast solution, a unicast-based approach in general leads to more packets and to increased latency, with the exact values strongly depending on the topology. Of course, the experimental evaluation also reveals that a multicast-based solution is inferior in cases where packet loss cannot be tolerated, caching is relevant, security is required or heterogeneous groups need to be supported.

3.8 Conclusions/Outlook

In this paper we have presented a novel solution for interacting with a group of CoAP resources across multiple smart objects. It provides an interesting alternative to multicast-based solutions, which are challenging to realize in a constrained environment. It is also an alternative to application-based solutions, which simply program the required functionality. An EM, which can reside anywhere, turns groups of resources into entities. A strong point of our approach is that it nicely integrates the important aspects of entity management: creation, validation, usage and manipulation. At the side of the constrained devices, it requires no additional complexity, except optional support for profiles in order to realize more powerful validation. The introduction of entity profiles introduces a lot of flexibility and opportunities for further extensions regarding how entities should behave. We have implemented our proposal and demonstrated and validated its feasibility. We have also performed a detailed performance evaluation of our solution. We explored several aspects (overhead, timing, scalability, *etc*.) related to the creation, validation and usage in realistic sensor networks and compared it with existing multicast-based solutions. As such, we think that our solution is a powerful enabler for group communication in LLNs and an interesting building block for Internet of Things (IoT) applications.

As future work we would like to test our solution on larger test-beds and with different use-cases. We will add more intelligence to the EM to make it optimize the response time and network overhead. One such optimization could be the adaptation of the CoAP retransmission parameters for certain members based on the history of communication with these members. Also, additional ways to interact with entities, by extending the profiles, will be investigated.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 258885 (SPITFIRE project), from the iMinds ICON projects O'CareCloudS and a VLIR PhD scholarship to Isam Ishaq.

References

[1] N. Kushalnagar, G. Montenegro, and C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. RFC 4919 (Informational), August 2007. Available from: http://www.ietf.org/rfc/rfc4919.txt.

[2] *Constrained RESTful Environments (core)*, 2014. [Online; accessed 19 March 2014]. Available from: http://datatracker.ietf.org/wg/core/.

[3] *Constrained RESTful Environments (core) – Charter for Working Group*, 2014. [Online; accessed 9 March 2014]. Available from: http://datatracker.ietf.org/wg/core/charter.

[4] I. Ishaq, J. Hoebeke, F. Van Den Abeele, I. Moerman, and P. Demeester. *Group Communication in Constrained Environments Using CoAP-based Entities*. In Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on, pages 345–350. IEEE, 2013.

[5] Z. Shelby, K. Hartke, and C. Bormann. *Constrained Application Protocol (CoAP)*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-ietf-core-coap-18.

[6] W. Colitti, K. Steenhaut, and N. De Caro. *Integrating wireless sensor networks with the web*. Extending the Internet to Low power and Lossy Networks (IP+ SN 2011), 2011.

[7] A. Dunkels et al. *Efficient application integration in IP-based sensor networks*. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pages 43–48. ACM, 2009.

[8] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security*. RFC 4347 (Proposed Standard), April 2006. Available from: http://www.ietf.org/rfc/rfc4347.txt.

[9] Z. Shelby. *Embedded web services*. IEEE Wireless Communications, 17(6):52, 2010.

[10] M. Nottingham. *Web Linking*. RFC 5988 (Proposed Standard), October 2010. Available from: http://www.ietf.org/rfc/rfc5988.txt.

[11] Z. Shelby. *Constrained RESTful Environments (CoRE) Link Format*. RFC 6690 (Proposed Standard), August 2012. Available from: http://www.ietf.org/rfc/rfc6690.txt.

[12] Z. Shelby, S. Krco, and C. Bormann. *CoRE Resource Directory*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-shelby-core-resource-directory-04.

[13] C. Bormann and Z. Shelby. *Blockwise Transfers in CoAP*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-ietf-core-block-14.

[14] A. Rahman and E. Dijk. *Group Communication for CoAP*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-ietf-core-groupcomm-18.

[15] Z. Shelby and C. Chauvenet. *The IPSO Application Framework*, 2014. [Online; accessed 3 June 2014]. Available from: http://www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf.

[16] A. Keranen, M. Ersue, and C. Bormann. *Terminology for Constrained Node Networks*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-ietf-lwig-terminology-07.

[17] S. Lai. *Heterogenous quorum-based wakeup scheduling for duty-cycled wireless sensor networks*. PhD thesis, Virginia Polytechnic Institute and State University, 2009.

[18] J. Hui and R. M. Kelsey. *Tulticast Protocol for Low Power and Lossy Networks (MPL)*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-ietf-roll-trickle-mcast-04.

[19] G. Oikonomou, I. Phillips, and T. Tryfonas. *Ipv6 multicast forwarding in rpl-based wireless sensor networks*. Wireless personal communications, 73(3):1089–1116, 2013.

[20] M. Jung and W. Kastner. *Efficient group communication based on Web services for reliable control in wireless automation*. In Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE, pages 5716–5722. IEEE, 2013.

[21] Z. Shelby and M. Vial. *CoRE Interfaces*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-shelby-core-interfaces-03.

[22] H. Hasemann, O. Kleine, A. Kröller, M. Leggieri, and D. Pfisterer. *Annotating real-world objects using semantic entities*. In Wireless Sensor Networks, pages 67–82. Springer, 2013.

[23] C.-D. Hou, D. Li, J.-F. Qiu, H.-L. Shi, and L. Cui. *SeaHttp: A Resource-Oriented Protocol to Extend REST Style for Web of Things*. Journal of Computer Science and Technology, 29(2):205–215, 2014.

[24] S. Eswaran, A. Misra, F. Bergamaschi, and T. L. Porta. *Utility-based bandwidth adaptation in mission-oriented wireless sensor networks*. ACM Transactions on Sensor Networks (TOSN), 8(2):17, 2012.

[25] B. Greevenbosch, J. Hoebeke, and I. Ishaq. *CoAP Profile Description Format*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-greevenbosch-core-profile-description-01.

[26] *Introducing JSON*, 2014. [Online; accessed 11 March 2013]. Available from: http://www.json.org/.

[27] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester. *Facilitating sensor deployment, discovery and resource access using embedded web services*. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, pages 717–724. IEEE, 2012.

[28] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. *The Click modular router*. ACM Transactions on Computer Systems (TOCS), 18(3):263–297, 2000.

[29] *Zolertia Z1 Platform*, 2014. [Online; accessed 11 March 2013]. Available from: http://www.json.org/.

[30] *Contiki: The Open Source Operating System for the Internet of Things.*, 2014. [Online; accessed 26 March 2013]. Available from: http://www.contiki-os.org/.

[31] *Erbium (Er) REST Engine and CoAP Implementation for Contiki.*, 2014. [Online; accessed 26 March 2013]. Available from: http://people.inf.ethz.ch/mkovatsc/erbium.php.

[32] G. Oikonomou and I. Phillips. *Stateless multicast forwarding with RPL in 6LowPAN sensor networks*. In Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on, pages 272–277. IEEE, 2012.

[33] C. Lerche, K. Hartke, and M. Kovatsch. *Industry adoption of the internet of things: a constrained application protocol survey*. In Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on, pages 1–6. IEEE, 2012.

[34] F. Van den Abeele, J. Hoebeke, I. Ishaq, G. K. Teklemariam, J. Rossey, I. Moerman, and P. Demeester. *Building embedded applications via REST services for the Internet of Things*. In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, page 82. ACM, 2013.

[35] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester. *The w-iLab. t testbed*. In Testbeds and Research Infrastructures. Development of Networks and Communities, pages 145–154. Springer, 2011.

4

Observing CoAP Groups Efficiently

"Everything that happens happens as it should, and if you observe carefully, you will find this to be so."

– Marcus Aurelius (AD 121 - 180)

Individual CoAP resources can be observed, upon which interested clients are informed about any state changes. Our group communication solution presented in Chapter 3, enables the interaction with a group of resources by exposing the group as a novel CoAP resource at the Entity Manager. A straightforward, but more difficult to answer, question is whether observations can also be applied to such CoAP group resources. Therefore, in this chapter, we build upon the CoAP group communication solution that was presented in Chapter 3 and extend it with such observe functionality. This functionality enables interested clients to observe state changes of the whole group, instead of observing state changes of the individual members and processing these changes at the clients side. Meanwhile CoAP and the CoAP group communications Internet drafts have been finalized and published RFCs. CoAP Observe was still an Internet Draft, but already in its final stage before becoming a standard.

⋆ ⋆ ⋆

I. Ishaq, J. Hoebeke, I. Moerman, and P. Demeester.

Published in Ad Hoc Networks Journal, Sep. 2015.

Abstract It is envisioned that by the year 2020 the Internet will contain more than 50 billion devices, among which the majority of them will have constraints in terms of memory, processing power or energy. As a consequence, they are often unable to run current standard Internet protocols, requiring special, optimized protocols. A number of these protocols, covering the different layers of the protocol stack, have been developed and standardized lately. At the application level, the Constrained Application Protocol (CoAP) is proposed by the IETF as an HTTP replacement that is suitable for constrained devices. CoAP is a very light-weight base protocol that can be extended with optional specifications to satisfy specific use case needs. Two important optional specifications are *observe*, allowing monitoring of a CoAP resource over a period of time, and *group communication*, supporting interactions with multiple CoAP devices at once. Currently, these two optional specifications do not work together, i.e., it is not possible to gain the benefits of both of them at the same time. In this paper we present an alternative and novel approach to CoAP group communication that works well with the CoAP observe extension. In addition, it enables to perform operations on the observed results, bringing intelligence closer to the data sources.

4.1 Introduction

The *Internet of Things* (IoT) is continuously expanding in many directions. It is expanding horizontally as new application domains start to rely on the opportunities offered by the Internet. At the same time, it is expanding vertically as within established connected domains, the types and numbers of devices connected to the Internet rapidly increase. Many of the newly connected things are connected using embedded devices that are typically optimized for low-cost and low-power consumption. These devices are constrained in their resources (CPU, RAM, ROM, etc.) and thus unable to run standard Internet protocols, which were originally designed with certain expectations of the devices' resources in mind. The local networks that connect these constrained devices together are often referred to as *low power and lossy networks* (LLNs).

In the last few years, many efforts have been put into the extension of standardized Internet technologies to constrained devices. Meanwhile, open standards exist to cover the complete networking stack and thus enabling, among many other things, compatibility between devices of various vendors [1].

Most noteworthy standardization efforts targeting the extension of Internet protocols toward constrained devices are the efforts of the *Internet Engineering Task Force* (IETF). Initial IETF efforts focused on the networking layers and resulted in the integration of constrained devices and LLNs in the IPv6 Internet. Later, the IETF started targeting the efficient integration of constrained devices in web services and therefore established the *Constrained RESTful Environments* (CoRE)

working group. CoRE aims to allow access to constrained devices in a RESTful way, similar to how most information on today's Internet is accessed over HTTP. CoRE takes as protocol design requirements the communication needs of *machine-to-machine* (M2M) applications such as smart energy and building automation [2].

So far, the main achievement of CoRE is the standardization of the *Constrained Application Protocol* (CoAP), which the IETF sees as an embedded counterpart of HTTP [3]. Constrained devices are turned into embedded web servers that make their resources accessible to Internet clients via the CoAP protocol. Typically, each of the constrained servers has at least one CoAP resource that may be queried by clients to obtain information about the devices themselves (e.g., battery level), about the environment that they monitor (e.g., temperature of the room), or to trigger the devices to perform real-world actions (switch the light on). These CoAP resources are identified by a *Uniform Resource Identifier* (URI) such as `coap://[aaaa::1]/temperature`.

As various IoT applications may have different communication requirements and the devices may be very heterogeneous in their capabilities, CoRE designed CoAP in way that the base protocol is kept as efficient and as simple as possible so it would run on even the most constrained devices. Features that might not be needed by all devices are standardized in a set of optional separate specifications. Two of the most important CoAP optional specifications are:

Group communication for CoAP. Section 8 of the base CoAP specification targets the well-recognized need for addressing several CoAP resources as a group, instead of addressing each resource individually. For example, this might be used to turn on all the CoAP-enabled lights in a room with a single CoAP request triggered by toggling the light switch. CoAP group communication relies on IP multicast to deliver the CoAP request to all group members. RFC 7390 [4] is an experimental RFC that provides best practices to use CoAP multicast messages and defines an optional interface for group management.

Observing resources in CoAP [5], is currently an Internet Draft that allows clients to register their interest in receiving notifications from CoAP servers once there are changes to the values of the observed resource. By doing so the client does not need to continuously pull the resource to find out whether it has changed its value. This typically leads to more efficient communication patterns that preserve valuable device and LLN resources. For example, this optional CoAP extension might be used to monitor the temperature inside a refrigerated container.

Unfortunately, these two important optional CoAP specifications have not been

designed to work together, i.e., it is not possible to combine these two specifications to observe a group of resources, e.g., to observe the status of the room lights group. This means that if a client needs to always have an up-to-date status of a group, it would need to either continuously pull the group or observe the members of the group individually. In other words, the client loses either the benefits gained by observing or the benefits gained by grouping. Both cases are suboptimal. In the former case, a lot of unnecessary traffic will be generated. In the latter case, clients would need to have the knowledge about the individual group members and to maintain the observe relationship with all of them individually. Either way leads to more complex and less flexible client applications.

Generally, there are two main approaches to solve the problem of observing CoAP groups of resources. In the first approach, multicast is used for group communication and the observe draft will need to be adapted to be able to deal with it. The use of multicast has certain benefits such as reducing the number of packets needed to deliver the request to the members. However, multicast also has its limitations such as poor reliability and being cache-unfriendly. In the second approach, multicast communication is avoided and other means for group communication are used and thus it becomes possible to observe resources in accordance with the observe draft. In this paper we examine the second approach by building up on our previous work [6], in which we have proposed an alternative method for CoAP group communication and submitted it as an Internet Draft [7]. Our method uses an Entity Manager (EM) acting as an aggregation point for the groups and offering CoAP resources that allow clients to create and interact with CoAP groups using unicast messages. In order to provide an alternative way for observing groups of resources, we present in this paper a major extension to our group communication solution [6], namely the efficient support for CoAP observe at the level of a group.

In this paper, we also propose to extend the *Entity Operations* concept, which we briefly introduced in [6], in order to support operations on the values of observed resources. By using Entity Operations it becomes possible to further reduce the communication needs between the group and the observing client(s). This efficient way of observing a group of resources is a way of providing distributed intelligence and is in line with current trends in research and industry to bring processing closer to the data sources [8, 9]. Our approach to observing a group of resources is also in the spirit of CoAP, i.e., processing of data and exposure as web services.

In summary, we present the following novel contributions in this paper:

- The observation of groups of CoAP resources, carefully managed by an EM, thereby pushing changes in a group or changes in processed information to the observing client(s).

- The possibility of applying operations to the observed resources within a

group (Entity Operations), further reducing the amount of communication.

- An improved architecture that enables to easily plug in support for new Entity Operations.
- Fine-grained control over the precision and frequency of notifications a client wants to receive about a group by assigning properties to the Entities (optimization techniques)
- An analysis and evaluation of the proposed optimization techniques.

The remainder of this paper is organized as follows. Section 4.2 explores the related work on CoAP group communication and the observation of groups. CoAP and a few needed optional specifications are explained in Section 4.3. In Section 4.4 we first summarize our previously introduced solution and then describe in detail how we extended it to support observation of groups. Next, in Section 4.5, we present a few optimizations to reduce the communication overhead of observing CoAP groups. In Section 4.6 we present our implementation and evaluate the functionality and the performance of our solution. Finally, Section 4.7 concludes this work with a summary and outlook.

4.2 Related work

As mentioned in the introduction, the basis of our work are the standardization efforts of the IETF CoRE Working Group, i.e., the CoAP [3] and two of its most important optional specifications: Observing Resources [5] and Group Communication [4].

The first optional specification specifies an observing mechanism that allows avoiding the continuous polling of CoAP resources by letting clients register for receiving notifications upon changes in the resources' values. Although this optional specification was still an Internet Draft at the time of writing this article, this draft has undergone many revisions by the Working Group and is expected to be approved without major changes.

The second of those two optional specifications was developed to address the group communication needs in CoAP and has been published as the experimental RFC 7390. This RFC specifies how CoAP should be used in a group communication context. It provides an approach for using CoAP on top of non-reliable IP multicast. Certainly, the use of multicasts allows reducing the amount of requests in the LLN, by sending one request to several destinations at the same time. However, this method for CoAP group communication is not supported with CoAP observe, since the latter only allows unicast messages. In addition, the use of multicast has other limitations such as being not cache-friendly and not supporting secure communication (see Section 4.3.3).

An earlier draft version of Group Communication for CoAP was the basis for the work presented in [10], in which web services based CoAP multicasts were used to access data from *Building Automation Systems* (BAS). It shows how using multicasts allows creating basic building control scenarios without the need of a central control unit. Certainly this approach has several advantages such as eliminating the need for a control unit, often a lower power consumption than using unicasts and its suitability in many non-critical use cases (due to the lack of reliability of multicasts). However, since this approach is based on IP multicast, it does not support CoAP Observe and exhibits the other limitations of multicasts as discussed in Section 4.3.3.

The authors of [11] present a lightweight multicast forwarding solution specifically designed for the requirements of service discovery in LLNs. Since in such cases group management is not required, the authors designed a simple alternative to multicast by relying on filtered flooding techniques to broadcast messages to all nodes in the local network while avoiding the creation of forwarding loops. The authors argue that broadcasting in ContikiMAC has certain drawbacks such as higher than necessary number of transmitted packets and lack of reliability. In order to tackle these issues, the authors replaced local broadcast with multiple unicast transmissions. By doing so, the number of transmitted packets, the packet propagation delay in the network and the reliability were improved at the cost of a slightly increased footprint. This approach is similar to ours in the way it relies on unicast rather than multicast to achieve group communication. However, unlike ours, this approach is suitable only for the cases where messages need to be sent to all nodes in the network.

In a M2M context, group communication is often more than just the communication aspect. If a machine receives data from multiple sources, the machine needs to know more details about the data it is getting from the various sources in order to be able to process this data correctly. This includes the need for strict typing of data to be able to understand the contents of the resources. Several initiatives that provide guidelines and standards in this regard exist. For example, the *CoRE link format* [12] provides a way to specify a *Resource Type* by using the `rt` attribute. In our work we rely on those standards to figure out how to combine different resources together. Combining notifications and performing operations on them before notifying the observer brings data processing closer to the data sources. Lately, several works in the industry and research community focus on distributed intelligence in the IoT field and in bringing data processing closer to the data sources (Fog computing). A major driver for these works is that connectivity to the Cloud is not infinite and not always cheap [8, 9, 13]. In that respect, our work can be seen as an innovative application enabler that can be supported by the Fog, which is one of the research areas identified in [13].

An approach somewhat more similar to ours, also using the notion of an entity

and supporting CoAP Observe, has been presented in [14]. The aim here is to annotate real-world objects by using entities that are automatically created based on semantic information, which resides on the constrained devices. One problem of using semantics on constrained devices is that semantics are verbose and can easily require a lot of memory. This makes them difficult to fit on constrained devices without applying compression techniques [15]. Further, in our approach users can create entities as required and we address important aspects related to entity validation and behavior.

The authors of [16] present an extension to CoAP called SeaHttp that enables communication with a group of resources. Similar to our work, SeaHttp also uses unicasts to realize group communication. The authors propose to extend CoAP with two additional methods (BRANCH and COMBINE) to allow members to join and leave groups without the need for a separate group manager. This means that members should have the intelligence to know which group they should join/leave. Constrained devices will not have this intelligence, so similar to our solution, we believe that SeaHttp does need a "manager" to inform the devices so they can take appropriate actions. Furthermore, BRANCH and COMBINE can maybe reduce the number of messages; however, the trade-off is the need to implement a new mechanism. It is better to use an approach that can be plugged in into any existing network without major modifications (or at least not a modification to every node). The article does not discuss whether SeaHttp resources support CoAP Observe or if the use of caches will still be possible with them. However, should this be possible then also the caches should be extended accordingly. Finally, this approach does not have the flexibility we target, since group members have to be reprogrammed with the groups they should join each time the requirements of the user changes.

The authors of [17] propose a scalable in-network storage solution for sensor networks benefiting from the available memory of the smart objects. This solution is built on agents and can autonomously manage the system without humans in the loop. To achieve this autonomous management, the solution relies on CoAP multicast messages to build and manage hierarchical zones that follow the structure of the building in which the system resides. The authors propose to combine the observe and multicast group communication in order to be able to update more than one storage location at once, without the need for those to register with the resource individually. This is achieved by sending the notification to the multicast address of the storage. This solution demonstrates another use case that needs to combine observe with group communication, but takes a different approach than ours to achieve it.

To our knowledge, these are the only works that explore communication solutions for interacting with a group of CoAP-enabled constrained devices. Next to these, there exist other solutions to realize group-like communication in con-

strained environments without using CoAP. For example, the Message Queue Telemetry Transport (MQTT) protocol is another application layer protocol designed for constrained devices [18]. MQTT uses a topic-based publish-subscribe architecture, i.e., clients utilize the services of a *Broker* to subscribe to *Topics* and get all the *Messages* that are published to that topic. Unlike CoAP, MQTT relies on TCP as underlying transport protocol and thus inherits its reliability. While the base CoAP does not provide any *Quality of Service* (QoS), MQTT provides its own QoS mechanism. MQTT provides its own way of group communication with observe support, by allowing multiple publishers to publish to the same topic and by allowing multiple clients to subscribe to it. This can be seen as a form of group communication which exhibits some similarities with our proposed approach. However, it does not adhere to the REST principles that are commonly used in the Internet and, additionally, it does not provide the possibility to aggregate and manipulate notifications that are sent to the clients.

4.3 CoAP overview

The focus of this paper is to enable observing a group of resources from a service/application perspective in a way that is in line with existing and ongoing standardization activities in the field of IoT. In the last few years a lot of effort has been put in defining a standard application protocol, similar to HTTP, but more suitable for constrained devices, namely CoAP. The base CoAP protocol is defined in RFC 7252 [3] in conjunction with a number of additional specifications. In this section we briefly introduce the base CoAP specification and those optional specifications that are relevant to our group observing work.

4.3.1 Base CoAP – RFC 7252

The main idea behind designing CoAP was to create a protocol that uses the same RESTful model as HTTP, but it is much lighter so that it can run on constrained devices [19, 20]. The *Representational State Transfer* (REST) architecture uses a *request/response model* in which clients exchange representations of resources with servers. A client interested in the state of a resource initiates a *request* to the server, which then returns a *response* with a representation of the resource that was current at the time of the request. A *resource representation* either captures the current or the intended state of the resource.

In order to reduce the resource requirements of CoAP, it was designed to have a much shorter header and lower parsing complexity than HTTP. CoAP uses a compact (4-bytes) base binary header that may be followed by optional extensions. The CoAP interaction model is similar to the client/server model of HTTP. A client can send a CoAP request, requesting an action specified by a method code (GET, PUT,

POST or DELETE) on a resource (identified by a URI) on a server. The CoAP server processes the request and sends back a response containing a response code and an optional payload.

A message that does not require reliable transmission can be sent as a Non-confirmable Message (NON). This type of messages is not acknowledged and thus might get lost without the client and the server noticing it. Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport layer such as UDP and thus supports multicast requests. This allows the use of CoAP for point-to-multipoint interactions, which are commonly required in automation. Multicast CoAP requests are sent using NONs.

Since UDP does not provide reliable communication, optional reliability is supported within CoAP itself. This is done by using Confirmable Messages (CONs) to implement simple stop-and-wait retransmissions with exponential back-off.

To be able to offer communication needs that cannot be satisfied by the field of the base binary header alone, this base header may be followed by one or more of the following optional fields: Token (to correlate requests and responses), Options, and Payload. As an example of a simple CoAP option consider the `Max-Age` option. This option indicates the maximum time a response may be cached before it is considered not fresh. If this option is not included in any CoAP response, the client (or cache) should assume that the response will be fresh for 60 s and thus will not query it again within this period. Another example is the `Uri-Query` option, which is a string that specifies one argument parameterizing the resource. Clients can use this option for instance to request that the server should process the response in a different way.

In M2M applications where there are no humans in the loop, it is important to provide a way to discover resources offered by constrained servers. For HTTP Web Servers, the discovery of resources is typically called Web Linking – RFC 5988 [21]. The use of Web Linking for the description and discovery of resources hosted by constrained web servers (CoAP or HTTP) is specified by the CoRE Link Format – RFC 6690 [12]. This RFC defines a well-known relative URI `/.well-known/core` as a default entry-point for requesting a list of links to resources hosted by a CoAP server. Once the list of available resources is obtained from the server, the client can send further requests to obtain the value of a certain resource. The example in Figure 4.1 shows a CoAP client requesting the list of the available resources on a CoAP server (`GET /.well-known/core`). The returned list (in CoRE Link Format) shows that the server has, among others, a sensor resource called /s/t that, when queried, returns the temperature in degrees Celsius. The client then requests the value of this resource (`GET /s/t`) and receives a plain text reply from the server with the value of the current temperature as payload of the message (`23.5`).

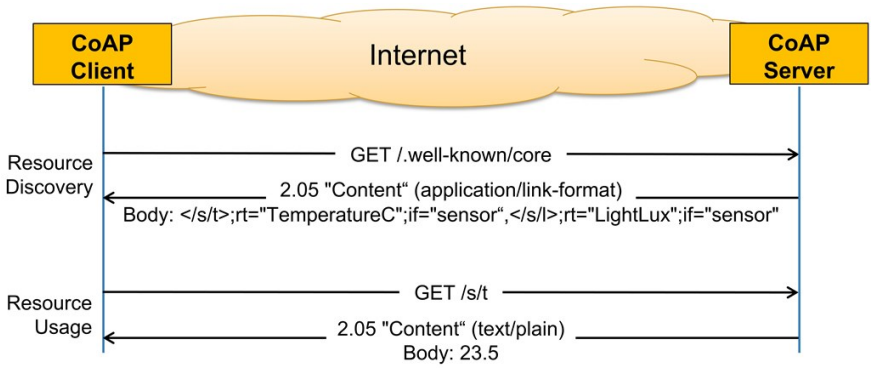


Figure 4.1: An example of Constrained RESTful Environments (CoRE) direct resource discovery and a Constrained Application Protocol (CoAP) request.

4.3.2 Observing resources

CoAP follows the REST model, i.e., clients initiate requests to servers in order to exchange representations of resources (see Section 4.3.1). A response returned by a server is a representation of the resource at the time of the respective request. The REST model does not work well when a client is interested in having a current representation of a resource over a time period. A few approaches exist for HTTP to solve this issue, e.g., repeated polling or HTTP long polling [22], However, these approaches generate significant complexity and/or overhead and thus are less applicable in constrained environments. In this subsection we describe the *Observing Resources in CoAP* optional extension that allows clients to register with servers their interest to receive notifications whenever a resource representation is changed. This extension keeps the architectural properties of REST but avoids the repeated pulling of resources.

4.3.2.1 Observe option

The *Observing Resources in CoAP* extension defines a new CoAP option and an associated protocol for its use. Thus, it extends the base CoAP protocol with a mechanism for a CoAP client to *observe* a resource on a CoAP server. When a client is interested in observing a resource over a time period, it adds the `Observe` Option to its request to the server. The server replies to the client with the current representation of the resource and updates the client about any changes to the representation as long as the client is interested in the resource.

Figure 4.2 shows an example of a CoAP client *registering* its interest in a resource and receiving two *notifications*: the first with the current state upon registration, and the second upon a change to the resource state. Both the registration request and the notification are identified as such by the presence of the `Observe`

Option. The client sets the value of the `Observe` Option to zero to indicate a registration request. In notifications, the server sets the value of the `Observe` Option to provide a sequence number that the client can use for reordering notifications. This number does not have to be sequential, but must be higher than the previous notifications. In order to make it possible for the client to correlate notifications with requests, the server must add the same token that the client used in the registration to all notifications.

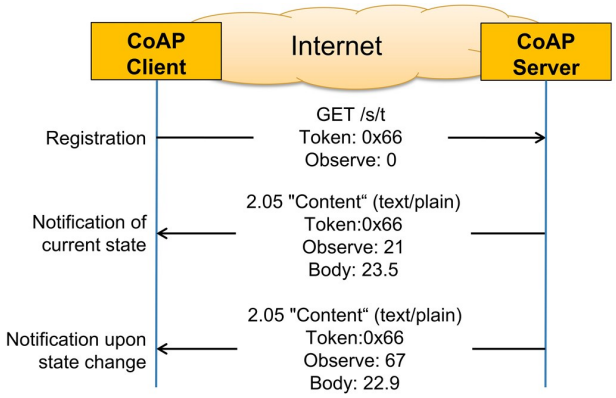


Figure 4.2: Observing a resource in CoAP. Clients register their interest in observing a resource. Servers then notify clients whenever the resource changes its value. The token must be the same in order to match the notifications to the registration.

4.3.2.2 Observe consistency model

The goal of the Observe protocol is to keep the resource state observed by the client as closely in sync with the actual state at the server as possible. However, due to signal propagation delay and the lossy nature of LLNs, it cannot be avoided that the client and the server become out of sync sometimes. Also, due to CoAP's congestion control mechanisms and the lossy nature of LLNs, clients cannot rely on observing every single state that a resource might go through.

While the Observe protocol can not guarantee that clients will always have an up-to-date representation of the observed resource, it follows a best-effort approach to get to this goal as close as possible. The protocol uses the `MAX_AGE` option (Section 4.3.1) to label notifications with a maximum duration up to which it is acceptable for the observed state and the actual state to be out of sync. The protocol is designed on the principle of eventual consistency. This means that, if the resource does not undergo a new change in state, eventually all observing clients will have a current representation of the latest resource state.

4.3.2.3 Observe extension to Web Linking

CoAP servers can indicate to the clients that a resource is observable using Web Linking. To do so, the server extends the resource's entry in `/.well-known/core` with a new attribute: `obs`. When this attribute is present (it should not contain a value) it is a hint that the resource is suitable for observation. For example, to hint that the resource is observable, the entry for `/s/t` from Figure 4.1 will become:

```
</s/t>;obs;rt="TemperatureC";if="sensor"
```

4.3.3 Group communication

The IETF CoRE working group has recognized the need to support a non-reliable multicast message to be sent to a group of devices to manipulate a resource on all the devices in the group. Therefore, they have included group communication in the base CoAP protocol and have developed the "Group Communication for CoAP - RFC 7390" [4], which provides guidance for how the CoAP protocol should be used in a group communication context. Group Communication refers to sending a single CoAP message to all members of a specific group by utilizing UDP/IP multicast for the requests, and unicast UDP/IP for the responses (if any). This implies that all the group members (the destination nodes) receive the exact same message.

The use of multicast is efficient in sending the requests, but does not affect the number of responses sent by the members since they are sent as unicasts. However, the use of multicasts has its limitations and challenges:

- The most prominent limitation is the **lack of reliability**, which makes it not suitable for all use cases.
- Another important limitation is the **cache-unfriendly** nature of multicasts preventing possible reduction of requests and replies by utilizing caches. Depending on the use case and network topology, the reduction of packets as a result of using a cache can be better than the reduction obtained from using multicasts.
- Also, multicasts are **not useful when a single user action needs to trigger different sensor requests**, since one multicast request delivers the same message to all group members.
- **Secure communication with the group members is not possible**, since all communication based on this RFC operates in CoAP NoSec (No Security) mode.
- **Multicast is not supported on all LLN MAC protocols**, especially MAC protocols that use *Radio Duty Cycles* (RDC) to shut down their radios when

not in use. For example, Xmac does not support multicast since it shuts down its receiver to avoid overhearing [23].

- Finally, and for our current work most important limitation, is that the **CoAP Observe Option does not support multicast** transport mode.

4.4 Flexible and observable entities

In [6] we have introduced our flexible unicast-based group communication solution for CoAP-enabled devices. In the following subsections we first summarize our previously introduced solution and present its improved architecture (Section 4.4.1) and then show how we extended it to support observation of groups (Section 4.4.2).

4.4.1 Group communication using unicasts

Our aim was to create an intermediate level of aggregation to be able to easily manipulate a group of resources across multiple smart objects. To avoid increasing the footprint of the constrained devices, we used the same technology as used to manipulate individual resources, i.e., CoAP, and extended it accordingly. Such a group of resources are called an *entity* and the resources themselves are called the entity *members*. An entity can be created, used or manipulated through a single CoAP request. When a client requests the creation of an entity, the EM performs a complete *validation* of the entity.

Furthermore, we have introduced in [6] the notion of profiles for the created entities. The use of entity profiles allows the client to specify in more detail how the entity should behave (e.g., if it should use confirmable or non-confirmable CoAP messages), and, through updating the profile, allows manipulation of this behavior. As such, we combined ease of creation, ease of usage and flexibility in behavior into a complete solution for interacting with CoAP resources from different objects inside an LLN. By building upon standardized concepts, the impact on the constrained devices was limited.

4.4.1.1 System overview

We call the component that manages the entities, the EM. This component, which can reside, e.g., on the Border Gateway of the LLN, is responsible for maintaining entities that are created from groups of resources residing on CoAP servers (i.e., sensors and actuators) inside the LLN. Clients on the Internet can interact with an EM to create new entities and/or customize how these entities should behave. Optionally clients can elect to contact a resource directory [24] in order to discover

which resources are available in the network. Figure 4.3 shows an overview of the involved components.

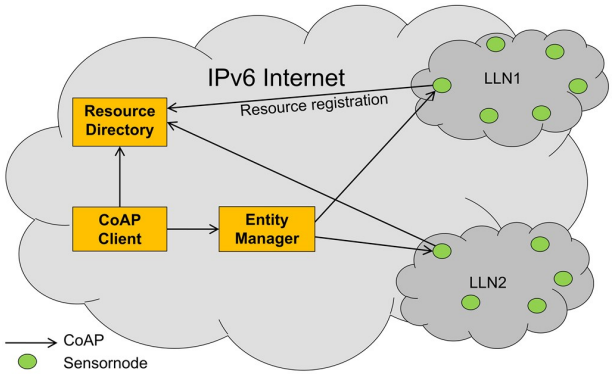


Figure 4.3: Clients create entities consisting of several smart object resources on the Entity Manager. Clients can optionally query a resource directory to discover the existence of the resources.

The EM functionality does not have to reside on a dedicated device. Theoretically, any CoAP server can be extended to become an EM (Figure 4.4). The choice of the most appropriate location to put the EM functionality depends on the size and topology of the network. For example, it can reside on a smart object in the constrained network with enough resources, in the Cloud, on the client device itself, or on a gateway at the edge of the LLN. The latter case has the added benefit that security can be centrally managed besides offloading the processing from constrained devices. One can also decide to implement multiple EMs (at the same or at different locations) to avoid having a single point of failure and thus improving reliability, availability and scalability.

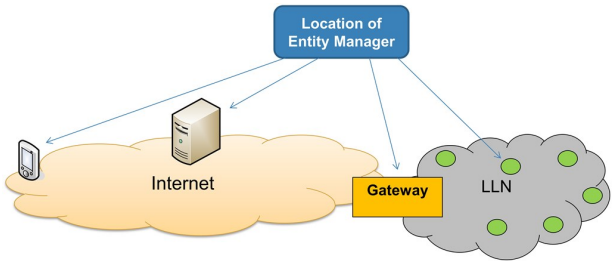


Figure 4.4: The Entity Manager (EM) functionality can be integrated into any CoAP server. The optimal location for the EM depends on the use case.

Regardless of the location of the EM, it will serve as a proxy between the client and the constrained devices. Client requests will be sent to the EM, which

will analyze and verify the requests and then issue the appropriate requests to the constrained devices using CoAP. Once the EM receives responses from the constrained devices, it will combine them according to the needs of the client and will send back an aggregated response to the client.

When a client tries to create a new entity consisting of a group of resources inside LLNs, the EM performs a sanity check on the request in order to make sure that the resulting entity would make sense. We call this sanity check the *Entity Validation*. For example, it verifies that the resources inside the entity are valid, whether they support a certain content format and whether their data can be aggregated. Customization of the entity behavior is accomplished by creating *profiles* for the entities. A profile of an entity can specify for example whether to return the values of all resources in the entity, only the computed average of all values or a subset of all values. Figure 4.5 shows a high-level structure of the EM. It shows that the EM contains three databases:

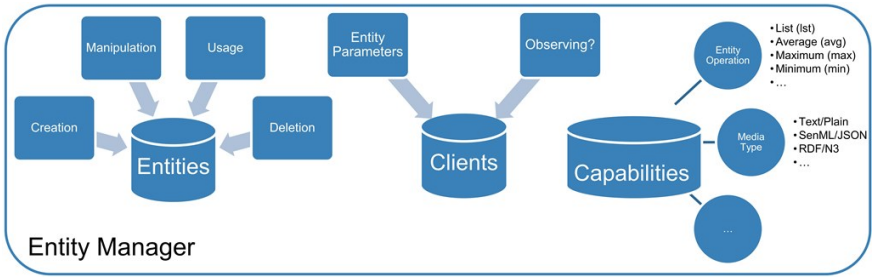


Figure 4.5: The EM manages the live-cycle of entities from creation until deletion and manages the relationship to the clients that are using the entities. In addition, it contains a knowledge database about the capabilities it can use to match user requests with sensor capabilities.

Entity Database: In this database, all entities are stored along with their profiles as defined by the user.

Clients Database: In this database, the EM stores the details of the clients that it is currently serving. These details include any customizations of the entity properties for the particular client and whether this client is observing the entity. If the client is no longer observing the entity, the EM removes the client details from the database once the EM sends a reply to the client. However, if the client is observing the entity, the EM keeps the client details until it stops observing.

Capabilities Database: This optional database provides rules and knowledge that the EM uses to match user requests with sensor capabilities. This can be as simple as translating a request for temperature in degrees

Celsius while obtaining the data from a sensor that only supports Fahrenheit. It can also be more complex, e.g., converting resource representations from one content format into the other. It also provides a library of operations that can be applied on the individual member responses in order to create the aggregated response sent to the client.

4.4.1.2 Entity creation

To facilitate the creation and manipulation of entities, the EM offers a CoAP resource "/e". We call this resource the Entity Management Resource. This interface only supports the CoAP POST request method. As payload of the request, it expects a collection of resources in CoRE link format [12], which together should form the entity. In the response, the `Location-Path` CoAP option is used to specify the name of the newly created resource. In the current design, the payload of the response is in plain text and describes the results of the validation tests performed by the EM on the collection of resources.

Thus, when a client wants to create an entity consisting of several members, it has to compose a CoAP POST request and send it to the Entity Management resource on the EM. The EM creates the entity, assigns it a unique URI, and stores the entity in the entity database for future usage. Then the EM starts the entity validation process (see [6] for details). The client is informed about the URI to use in order to access or further customize the newly created entity and about the results of the validation of the entity.

An example of the entity creation process is shown in Figure 4.6. In this simple example the client requests the creation of an entity consisting of two members: `coap://[Sen1]/rh` and `coap://[Sen2]/rh` , with `Sen1` and `Sen2` the IPv6 addresses of the two sensors. The EM creates the new entity, assigns it the URI `/1` and informs the client about the newly created entity. From now on, any client can access the newly created entity by accessing the `/1` resource on the EM. Please note the validation process is not shown in Figure 4.6 for simplicity.

At creation time, the client can use optional URI-Query CoAP options with the POST request to specify the name of the entity to be created or to customize the default behavior of the entity. For example, to create the entity "room_humidity" that returns by default the minimum value of all members when queried, a POST to `coap://[EM]/e?path="/room_humidity"&eo="min"` is needed. We will discuss customization of the entity behavior in more detail in Section 4.5.

4.4.1.3 Entity usage

When a client wants to use an existing entity, it sends a standard CoAP request to the entity resource on the EM. This request may include optional Uri-Queries

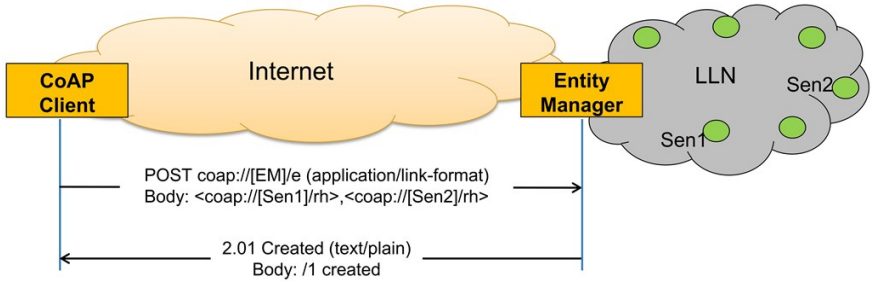


Figure 4.6: A CoAP client requesting from an EM to create a new entity that contains two resources.

to customize the behavior of the entity for this particular request. Simplified, i.e., ignoring all error conditions, the EM handles entity usage requests as shown in Figure 4.7. First, the EM analyses the request in order to obtain the set of individual requests it needs to send to the entity members. For this, information residing in the Clients Database and Entities Database is being used. Next, it creates the required requests and schedules them for delivery to the members. Each request passes then through a cache, which decides whether to reply directly (if the cache has a fresh reply) or to actually pass the request to the member. Either way, the EM eventually receives a response for its request and stores it in its temporary responses storage. Once the EM has received all required responses, it uses its capabilities database in order to convert all replies to a common format and then applies the needed Entity Operation to obtain a single response and send it to the client.

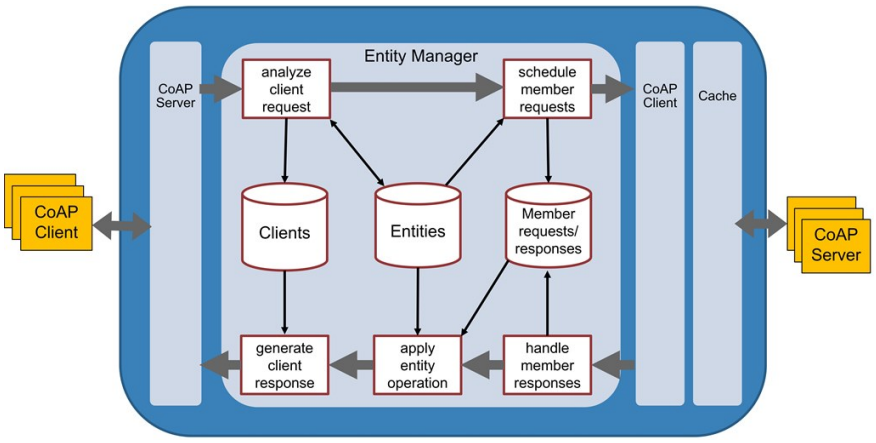


Figure 4.7: Handling client requests to use existing entities.

Figure 4.8 shows an example of using the entity that was created previously in Figure 4.6. The client issues a GET request on the entity's resource `/1`. This results in the EM issuing two GET requests to the individual members, waiting for replies from both of them and then sending both results in one combined response back to the client. In this case the reply from the EM to the client was in senml+json format [25]. This format is considered simple enough to be constructed by constrained devices, and at the same time to be parsed efficiently by servers.

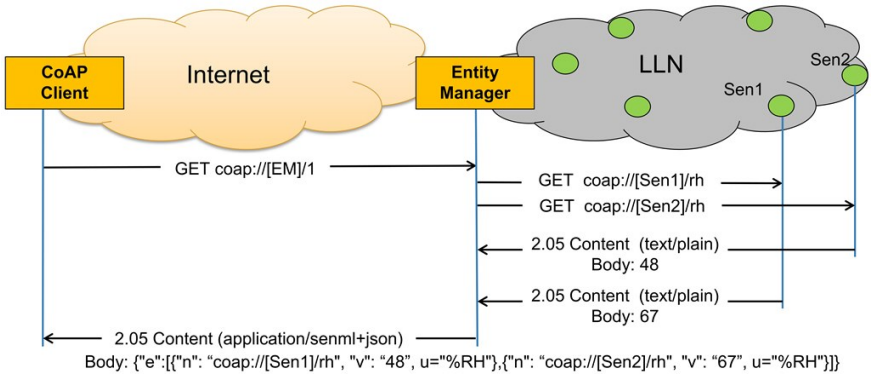


Figure 4.8: A CoAP client requesting from an EM to obtain the values for the entity that was previously created in Figure 4.6.

The client can decide to query the entity using its default behavior as described in the entity profile or to customize its behavior. To customize the behavior the client can include URI queries in its request to the entity. An example of the supported URI queries that can currently be used is the Entity Operation (eo). For example, the client should use the URI: `coap://[EM]/1?eo="avg"` to obtain the average value of the two relative humidity sensors of the entity `/1` in Figure 4.8.

4.4.2 Group observation

When a client creates an entity, it can indicate to the EM that this entity should be observable by adding a Uri-Query `obs=1` to the entity creation request. In such cases, the EM extends the entity validation steps to check that all the entity members are indeed observable. Typically, this information is contained in the resource's entry in `/.well-known/core` (see Section 4.3.2.3). If all members are observable the entity is also flagged as observable by adding the `obs` attribute to the entity resource in `/.well-known/core`.

When a client is interested in observing the group, it establishes an observe relationship with the entity resource in the same way it would start an observe

relationship with any other observable resource, i.e., by sending a GET request with the `Observe` Option set with a value of zero. The EM adds the client to its *list of observers* and in turn starts observing all the members of the group in the same manner.

When an observed group member changes its value, it sends a notification to the EM informing it about the new value. The EM manager stores the new value and potentially notifies the client about the new change either immediately or after a certain time. In the remainder of this article, we call the notifications from entity members to the EM *member notifications* and call the notification sent from the EM to the clients *client notifications*. The decision whether to notify the client and when to do so is based on three configurable entity properties that have default values set at the time of creation of the entity. If needed, these values can also be changed on a per request basis by the client at the time of starting the observe relationship. These properties are the *Notifications Aggregation Window*, the *Entity Operation* and the *Entity Precision*. We will explain these properties in detail in Section 4.5.

When the client is no longer interested in observing the group, it can terminate the observe relationship with the entity resource in the same way it would terminate an observe relationship with any other observable CoAP resource. The EM in turn then stops observing all the group members in the same way and removes the client from the list of observers.

4.5 Optimizing the number of client notifications

In the previous section, we have described our solution that allows CoAP clients to observe a group of CoAP resources by issuing a single CoAP observe request to the entity resource at the EM. This approach not only makes it easy to start observing several resources, but also opens new possibilities to combine and manipulate notifications at the EM before sending them to the observer. This has the benefit that intelligence is brought closer to the data sources, which helps reduce network traffic and improve its responsiveness. Consider the simple topology shown in Figure 4.9 consisting of an LLN connected to the Internet via a gateway, which is also acting as the EM. At the EM, an entity has been defined, which is being observed by three different clients that have different capabilities and different types of Internet connection and that are observing the group for different purposes.

The Internet server has a fast Internet connection and needs to store all the changes that occur to all members of the group.

The smartphone is connected to the Internet via an expensive data package and needs to be notified only when certain changes occur to the group as such.

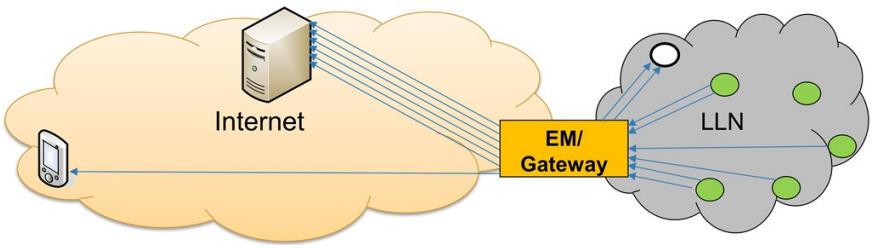


Figure 4.9: When an observed group member notifies the EM that its resource value has changed, the EM decides if it should notify observing clients about this change based on a set of per client configurable entity properties. This way, clients can elect to be notified about all changes of the individual group members, or just when certain changes occur to the group as such.

The smart object in the LLN is a constrained device containing an actuator that needs to be adjusted according to the status of the complete group.

In the simplest way of observing an entity, the EM manager acts as a forwarder of notifications. That is, the number of client notifications sent to the client (client notifications) will be the sum of all the notifications that the EM has received from the group members (member notifications). This will satisfy the needs of the Internet server of knowing all the changes that occur to the individual members. However, this is clearly not what the smartphone and the smart object need to know and possibly could handle. In order to satisfy the needs of these observing clients, the EM should be able to offer ways that allow the clients to specify what types of notifications they are interested in receiving. In this section, we explore some techniques that we have added to the EM to achieve this goal. For each technique we derive a simple mathematical model that helps understand the benefits of using this technique. These techniques can be applied individually or combined in order to achieve even better optimization of the client notifications. In Section 4.6.3 we will validate these models experimentally.

4.5.1 Entity operation

When a client observes a group, there are cases where the client is interested in observing all changes triggered by the group members. However, there are also certain cases where the client only needs to know about changes of the group status as such. For example, an environmental monitor might be interested in observing the maximum temperature of all sensors in a large cool storage room, in order to raise an alarm if this value exceeds a certain threshold. In order to be able to let the client observe the group according to the client's needs, the EM allows clients to select between different Entity Operations when they register their entity observation requests. Clients can do so by adding the Uri-Query `eo={lst,avg,min,max,`

. . . } to the observe registration requests. The default Entity Operation `lst` notifies the observer with a list of values of all entity members each time the EM gets a member notification. The other Entity Operations perform arithmetical operations on the individual member values. For this the EM needs to be able to understand the data formats, so strict typing needed. At this moment this is the responsibility of the entity creator, but the EM architecture allows automating this during the validation of the entity and during its usage.

Different clients can observe the same entity using different Entity Operations and thus might get different notifications from the EM for the same change in the observed entity. Consider again the topology shown in Figure 4.9, where the cloud server might observe using `eo=lst` and will thus be notified about all changes in the group. On the other hand the smartphone is using `eo=max` and will thus be notified only if the maximum value of all members changes. The latter case might help reduce the cost of mobile communication and reducing the power consumption of the smartphone in processing not needed information.

In order to calculate the savings in the number of notifications as a result of using Entity Operations, let us assume that the client is observing an entity with g members and thus the EM is observing all of its g members. At each sampling point, every member might change its value with a probability p_{change}. If the value changes, the EM is notified, which in turn might notify the client, based on the selected Entity Operation. The number of member notifications that the EM receives n_m is the sum of all notifications sent by the individual members:

$$n_m = s \times \sum_{i=1}^{g} p_{change_i} \tag{4.1}$$

where s is the number of sampling points. For simplicity, we assume that all members have equal probability of value change and thus

$$n_m = s \times g \times p_{change} \tag{4.2}$$

In the case of using `eo=lst`, the EM sends a notification to the observing client each time it receives a notification from any member. In this case, the number of client notifications n_{lst} equals the number of received member notifications n_m and the ratio r between client and member notifications is simply

$$r_{lst} = \frac{n_{lst}}{n_m} = 1 \tag{4.3}$$

In the case of using `eo=max`, the client is interested only in the maximum value of all observed members and thus the EM notifies the client only when the maximum value changes. This means that the EM sends notifications to the client in the following cases:

- The member with the current maximum value (probability: $1/g$) changes its value (probability: p_{change}).
- A member that does not currently have the maximum value (probability: $(g-1)/g$) changes its value (probability: p_{change}) and the new value is higher than the current maximum (probability: p_{higher})

On the other hand, the EM does not send notifications to the client when a member that does not currently have the maximum value changes its value and the new value is still lower than the current maximum. For the sake of simplicity, we neglected the cases in which more than one member have the exact maximum value. Accordingly, it is expected that the number of client notifications n_{max} is less than the number of received member notifications n_m and we can calculate n_{max} as follows:

$$\begin{aligned} n_{max} &= s \times g \times \left(\frac{1}{g} \times p_{change} + \frac{g-1}{g} \times p_{change} \times p_{higher} \right) \\ &= s \times p_{change} \times (1 + (g-1) \times p_{higher}) \end{aligned} \tag{4.4}$$

We can calculate the ratio between client and member notifications as follows:

$$r_{max} = \frac{n_{max}}{n_m} = \frac{1 + g \times p_{higher} - p_{higher}}{g} \tag{4.5}$$

This equation means that the ratio r_{max} between client and member notifications when using the `max` Entity Operation is independent from p_{change}. Moreover, r_{max} drops exponentially with the increase of the group size g and drops linearly with the drop of the probability that a new member notification's value is higher than the same members previous notification's value p_{higher}. These relationships are visualized in the 3D plot in Figure 4.10.

4.5.2 Entity precision

When the EM applies the Entity Operations to aggregate the member notifications that contain sensor values into a single client notification, it uses the default *Entity Precision*, which depends on the Entity Operation itself. The Entity Precision specifies the number of digits after the comma for real numbers – i.e., a precision of two results in numbers with two decimal digits whereas a precision of zero results in integer values. For example, for the Entity Operations `lst`, `max` and `min`, the EM sends the client notifications using the same precision as the one used in the member notifications. For `eo=avg`, the EM uses a precision of six decimal digits for the calculated average value.

In some cases, the client might not be interested in the full precision of the measurements obtained by the individual sensors. As such, it can instruct the EM

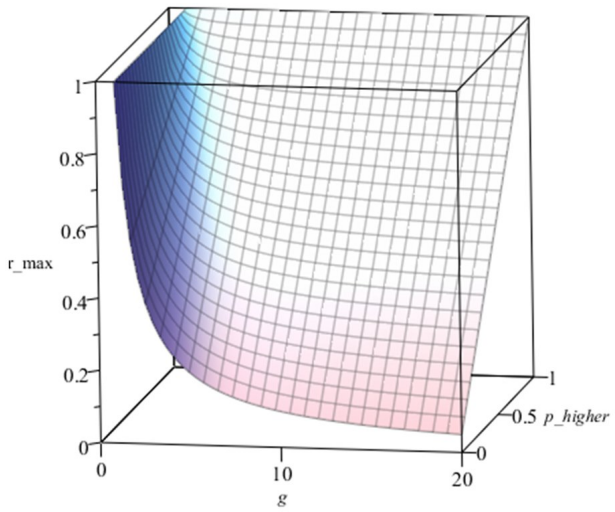


Figure 4.10: The ratio r_{max} between client and member notifications when using the `max` Entity Operation has a linear dependency on p_{higher} and a reverse exponential dependency on the group size g.

to send the results using a specific precision. When the client creates the entity or when it tries to use an existing entity, the client has the possibility to specify this precision for the values it receives from the EM by adding the Uri-Query `precision=x`, where `x` is the desired precision. If the client does not specify a precision, the EM passes the results to the client using the default precision.

To illustrate how reducing the precision can reduce the number of client notifications, please consider Figure 4.11. This figure is based on the temperature logs of Asheville regional airport for the first 12 hours of 2007.[1] It shows the notifications that would be generated as a result of the dew point temperature using two precisions: the original data precision (1 decimal point), and a reduced precision (integers only). In this particular example reducing the precision will result in 8 notifications instead of the original 16 notifications, i.e., 50% reduction in the number of notifications. Using a lower precision becomes even more relevant when applying Entity Operations that do not result in rounded numbers, e.g., when computing averages.

Consider again the topology shown in Figure 4.9. Whenever a client uses `eo=lst` with default precision, the EM sends a new notification to the observing client each time it receives a new notification from any member with the values received from the members in the same precision the EM received them. In this case, the number of sent notifications to the client n_{lst} equals the number of re-

[1] Data source: http://cdo.ncdc.noaa.gov/qclcd/qclcdhrlyobs.htm

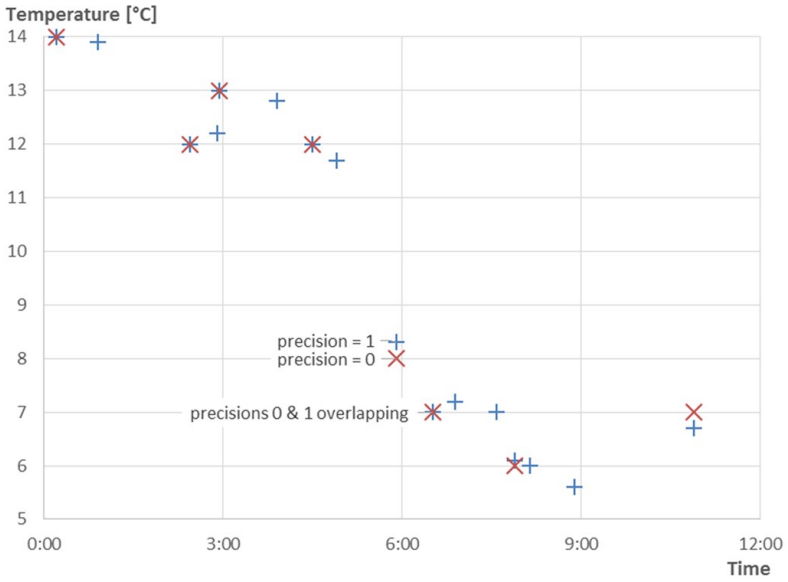


Figure 4.11: Reducing the precision of notifications can considerably reduce the number of client notifications.

ceived notifications from the members n_m and the ratio between sent and received notifications equals one – as calculated in Equation (4.3).

When the client specifies a precision for the entity that is lower than the precision of the individual members, the EM first rounds the individual values to the specified precision and notifies the client only if these rounded values have changed. Thus, the number of client notifications is expected to be less than the number of member notifications. Of course, selecting a higher precision than the individual members does not have any effect on reducing the number of client notifications. The amount of reduction in the number of client notifications depends on p_{same} which is the probability that the changed value by any given member is still the same as the last value, after rounding both values to a lower precision. The lower the precision is, the higher the p_{same} will be. Since the EM combines different members notifications together, the ratio of client to member notifications for a lower precision $r_{precision}$ is independent of the group size and can be calculated as:

$$r_{precision} = 1 - p_{same} \tag{4.6}$$

4.5.3 Notifications aggregation window

The third technique available for clients to reduce the number of notifications that EM sends to them is the use of a *Notifications Aggregation Window*. This entity property is set as the Uri-Query `obswin=w`, where `w` is the number of seconds that the EM can buffer member notifications in order to allow other member notifications to arrive at the EM before combining them together and sending a single client notification. All notifications from one or more members that arrive during this period are sent in an aggregated manner in a single client notification, once the period expires. Of course using the Notifications Aggregation Window has the disadvantage that member notifications are delayed – in the worst case up to value of the Notifications Aggregation Window itself. However, often this delay can be tolerated for the sake of the reduction of the number of notifications.

When the EM is not using a Notifications Aggregation Window ($w = 0$), then the ratio between client and member notifications r equals one. Every member notification results in a client notification. As w increases r exponentially drops and approaches 0 as w approaches infinity. In that case, the EM will wait infinitely and will no longer send any client notifications (Figure 4.12). Of course, this is a simplification as we have ignored the fact that at a certain moment, the aggregated information might no longer fit into a single packet. In that case, the notification needs to be fragmented into multiple packets (e.g., CoAP blocks) or multiple notifications, each fitting into a single packet, need to be sent. For simplicity, this behavior is ignored.

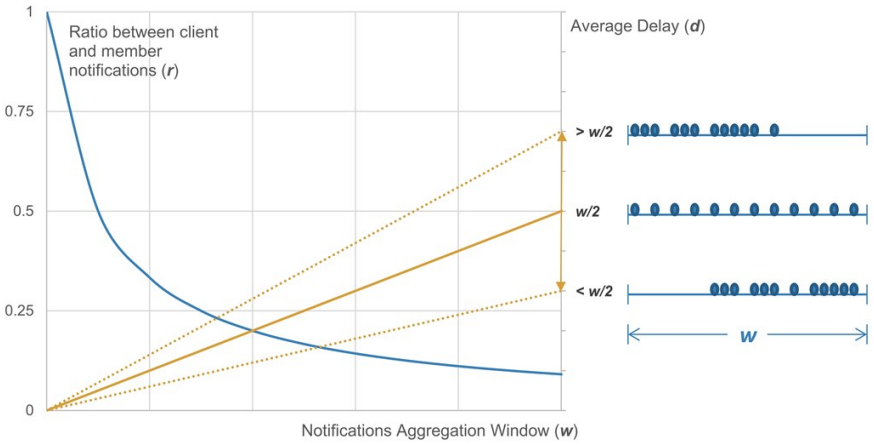


Figure 4.12: Ratio between client and member notifications and the average delay for member notifications as a function of the Notifications Aggregation Window.

If the member notifications are completely independent from each other, they will arrive randomly at the EM. Consequently, every member notification will be

delayed for a time d that is between 0 and the Notifications Aggregation Window w and the average delay $\bar{d}$ over all member notifications equals its half, i.e.,

$$\bar{d} = w/2 \tag{4.7}$$

In certain cases, when an environmental change occurs, this change is registered by many sensors in a relatively short period. This means that the member notifications are not completely independent, but have a certain correlation. This correlation will likely affect the distribution of the member notifications over the Notifications Aggregation Window. If the distribution of the arrivals tends to be concentrated more at the beginning of the window, the member notifications will be delayed on average longer than $w/2$. Similarly, if the distribution of the arrivals is more concentrated at the end of the window, then the average delay will become less than $w/2$ (see right side of Figure 4.12). Also, in both cases the Notifications Aggregation Window is larger than needed and is thus suboptimal. The optimal window size would be a window that is just large enough to include all correlated member notifications and this window should start the moment the first member notification of a batch of correlated member notifications arrives. Clients that observe the entity might have knowledge about the optimal window size, for example from knowing the specs of the used sensors and the properties of the observed environment. Alternatively, the EM may automatically adapt the window size based on the history of the distribution of notification.

In order to visualize the effect of using the Notifications Aggregation Window, we consider an entity with five members. In Figure 4.13 we show the effect of the size of the Notifications Aggregation Window (w) on the number of client notifications (n) and the average introduced delay (d) for two sample arrival patterns of member notifications. The figure contains two sets of five correlated member notifications that arrive at the EM with different distributions. The empty circles show when member notifications arrive at the EM and the filled circles show when client notifications are sent by the EM to the observing client. When $w = 0$, the EM notifies the observing client as soon as it receives a notification from the observed members. As $w = 0$ increases, n generally decreases to reach 1 when w is large enough to aggregate all member notifications into a single client notification. i.e., $w = 4s$ in both examples. Increasing w beyond 4 cannot optimize n any further; it just makes d worse. Although in the above examples the member notifications were generated by different members, in fact the number of notifications sent from the EM to the client does not change based on the source of the notifications. If a certain member sends more than one notification in the same Notifications Aggregation Window, the newest value will replace the older values. This means that the client might skip a change of that member if the change occurred within a period shorter than w.

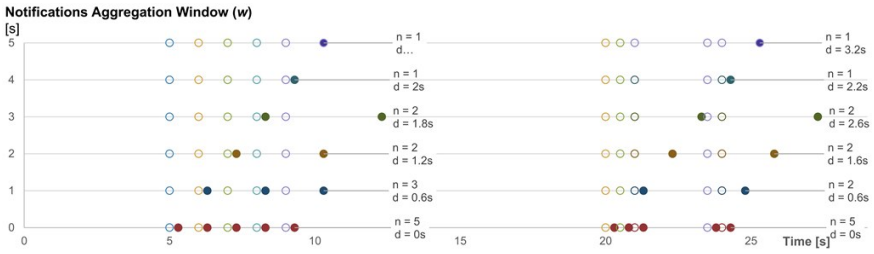


Figure 4.13: Client notifications generated by the EM as a result of receiving member notifications. The number of client notifications (n) and the average delay (d) depend on the number of notifications received, the Notifications Aggregation Window size (w) and on the distribution of the received notifications. Note: The empty circles indicate when member notifications are received by the EM; the filled circles indicate when client notifications are sent by the EM to the observer.

4.6 Implementation and evaluation

Our solution described above enables the use of unicast messages as an alternative to using multicasts for realizing CoAP group communication. By relying only on standard CoAP unicast messages, our solution is able to extend the CoAP observe option to be suitable for group communication as well. In order to evaluate our solution and to show how it can be used in a real-world scenario we have implemented it and built a demo box for demonstration purposes [26]. In this section we present the implementation of our solution followed by functional and performance evaluations.

4.6.1 Implementation

The key in our group communication solution is the EM. We have implemented the EM functionality on the gateway of the LLN using the CoAP++ framework [27]. The CoAP++ framework and the EM implementation on top of it have been realized in *Click Router*, a C++ based modular framework that can be used to realize any network packet processing functionality [28]. The CoAP++ framework is modular and can be used for various scenarios (Figure 4.14). Our EM needs to act as a CoAP client and as a CoAP server at the same time. The CoAP client functionality is needed to communicate with the entity members while the CoAP server functionality is needed to serve the clients of the entities. To realize those client and server functionalities, we use the *CoAP Client* and *CoAP Server* modules from the CoAP++ framework. The CoAP Server module functionality can be extended by developing add-ons to be plugged in to it. One such an add-on is the *Resource Directory*, which extends the framework with a CoAP Resource Directory. Our EM is also implemented as an add-on to the CoAP Server module. Furthermore,

since the CoAP++ framework already contains a *cache* module, we do not implement this functionality in the EM itself, but use the framework's cache module when it is needed. Finally, the CoAP++ framework includes a *Dummy Sensor* module (not shown in Figure 4.14 for simplicity). This module allows developers to define one or more software sensors in the framework. These software sensors offer several CoAP resources that can be queried and observed by CoAP clients. This way it becomes possible to perform initial testing against CoAP servers without the need for real sensors.

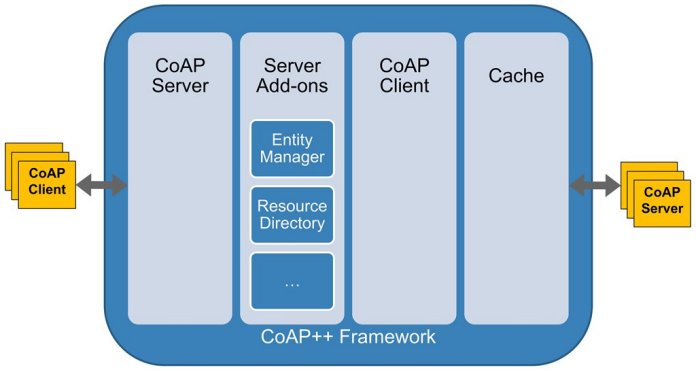


Figure 4.14: Location of the Entity Manager within the CoAP++ architecture

In addition to the software sensors from the Dummy Sensor module we have used Zolertia Z1's [29] and Rmoni RM090 [30] board as members. On these boards we have run a development version[2] of the popular Contiki operating system [31]. This version of Contiki was the most recent development version when we started our experiments. It supports multicast and includes a stable implementation of CoAP, namely the Erbium CoAP server [32]. In addition to providing base CoAP functionality, Erbium also supports the observe extension. Our group communication approach does not require any changes on the CoAP enabled constrained devices. However, in order to demonstrate how the EM can use resource profiles to validate entities, we have added resource profiles to the constrained CoAP servers.

4.6.2 Functional evaluation

The functionality for creating, validating, using and deleting observable entities has been implemented as described above. In this subsection we demonstrate the main functionalities of the implementation using a series of screen-shots covering

[2]Snapshot from the Contiki master branch on github on 2 July 2014

the life cycle of an observable entity (Figure 4.15). These screen-shots are taken using the CoAP++ client Graphical User Interface (GUI).

In Figure 4.15a the client requests to create an observable entity of three members. The members are three temperature resources on RM090 sensors. The EM validated the members by querying the profiles of these resources and validated that all the resources are observable by checking /.well-known/core of all sensors. The entity was created successfully and is ready for use since it passed the validation.

Next, the client checks the profile of the newly created entity (Figure 4.15b). The profile shows that the entity supports four Entity Operations: `lst, avg, min` and `max` whereas `lst` is the default operation since it is the first operation in the list of supported operations.

Next, the client starts observing the newly created entity without specifying any Uri-Queries and thus the default entity properties (Entity Operation, Entity Precision and Notifications Aggregation Window) are used. Figure 4.15c shows how the client is getting a notification containing all members values each time any member changes its value.

Since the client is actually interested in the average temperature of all three sensors, the client cancels the current observe relationship and starts observing the entity using the Entity Operation `avg` (Figure 4.15d). By doing so, the EM now calculates the average and only sends the calculated value to the client. If the average value does not change as a result of a change of one of the member values, no notification will be sent by the EM.

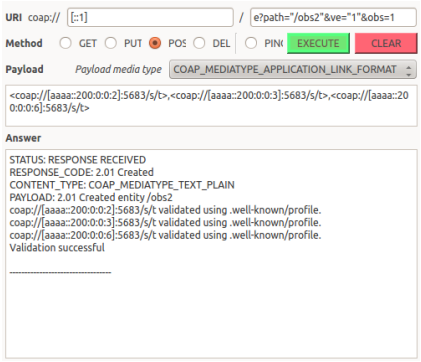
To complete the life cycle of an entity we show how the client deletes it in Figure 4.15e.

4.6.3 Performance evaluation

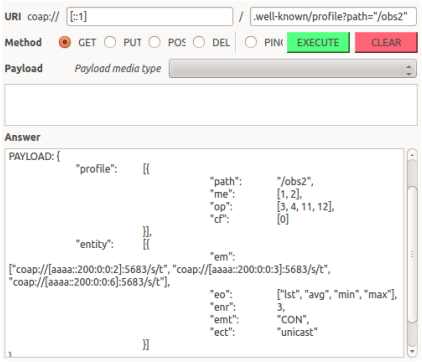
The three communication participants of our approach are the clients, the EM and the entity members. Clients might be constrained devices (e.g., a light switch triggering a group of lights) or not constrained (e.g., a building management computer). In any case, the use of our approach generally requires less processing and memory on the client side since complexity is outsourced to the EM (Fog idea). Clients need to worry about only one observe relationship with the EM, instead of maintaining observe relationships with all group members. Furthermore, clients need to store only final processed results, instead of storing intermediate results and processing them later on.

The EM implementation is highly dependent on the type of device on which EM is running. In our case, we used Linux, powered with click modular router and the CoAP++ Framework.

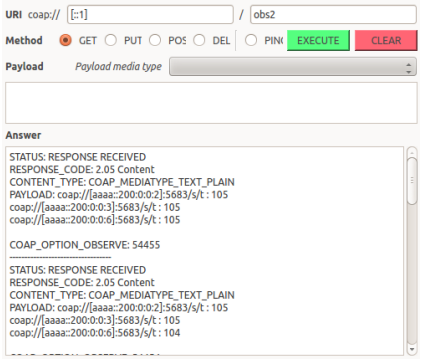
Our solution requires no additions to the CoAP implementation of the con-



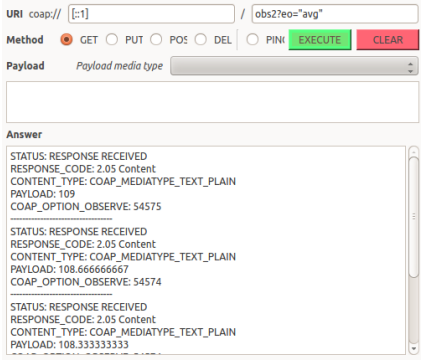
(a) Creating observable entity obs2.



(b) Profile of obs2.



(c) Observing with default properties.



(d) Observing with entity operation avg.

(e) Deleting the entity obs2.

Figure 4.15: Screen-shots using the CoAP++ client GUI to create, observe and delete an observable entity of three members.

strained devices acting as group members. Thus, it has no overhead in terms of memory and CPU. Since the EM behaves as a proxy between the clients and the constrained devices it also limits the number of observe relationships that constrained CoAP servers have to maintain. This not only helps to reduce the memory that should be reserved for maintaining the observe relationships, but also results in fewer packets sent by the constrained devices. This reduces energy consumption by the constrained devices, since a major source for it is radio transmissions, which is directly related to the number of notifications. The exact values for energy consumptions are hardware and MAC protocol dependent. Thus, in this paper we focus our evaluation on the number of notifications. For a discussion of the relation between the number of notifications and energy consumption, we refer to [33].

In Section 4.5 we have presented three techniques that clients, which want to observe a group, can use individually or combined in order to optimize the number of notifications they receive from the EM. For each technique we provided a simple mathematical model for the reduction in the number of client notifications. In this section we present the results of a series of experiments that we have conducted in order to validate these models. In those experiments we have used as entity members either software sensors from the CoAP++ Dummy Sensor module or RM090 sensors simulated in Cooja (the Contiki network simulator). In Table 4.1 we summarize the most important contiki and Cooja network simulator settings.

Table 4.1: Contiki and Cooja network simulator settings

Contiki	Version	Snapshot from the Contiki master branch on github on 2 July 2014
	Radio Duty Cycling (RDC)	ContikiMAC
	Media Access Control (MAC)	CDMA
	Routing Protocol	RPL
Cooja	Radio Medium	Unit Disk Graph Medium (UDGM): Distance Loss
	Transmit Ratio	100 %
	Receive Ratio	100 %

4.6.3.1 Entity operation

As described in Section 4.5.1 the Entity Operation (eo) property allows the client to select an operation that is applied on the values in the notifications from the entity members before sending the result the client. When using `eo=lst`, it is expected that the number of client notifications equals the number of member notifications regardless of the group size. For the other operations it is expected that the ratio between client and member notifications drops exponentially as the group size

increases. Please note that the exact speed at which the ratio drops depends on the selected entity operation and on the pattern of changes that occur to the individual members as can be seen in Equation (4.5) and Figure 4.10. This behavior can indeed be observed in Figure 4.16 for the three tested operations `lst`, `avg` and `max`.

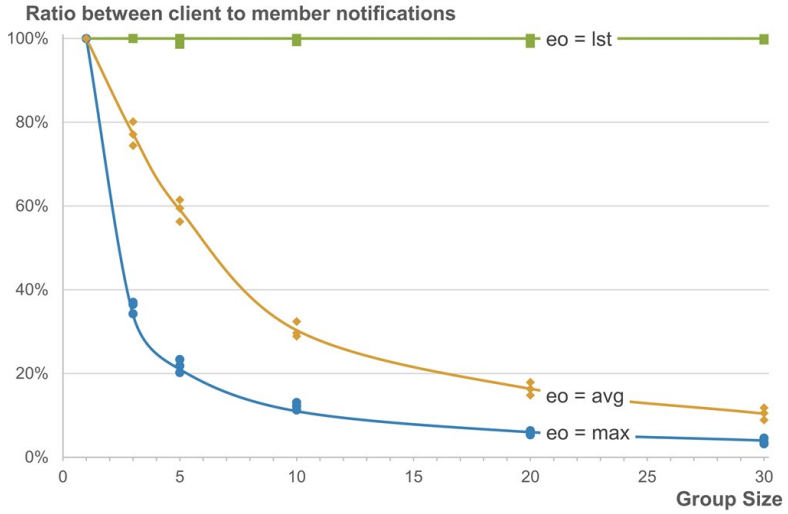


Figure 4.16: The ratio between the number of client notifications to the number of member notifications for different Entity Operations (lst: list all values; max: maximum value; avg: average value)

Each dot in Figure 4.16 represent an experiment while the continuous lines represent the average values. In these experiments we used software sensors as members. This has the benefit that it allows us to control the pattern in which the values of the sensors change over time and consequently the pattern of member notifications. In the experiments all sensors started from the same value. Every 5 s each sensor randomly decides if it wants to keep its current value, slightly increase it or slightly decrease it. This way we achieve that the values that we want to observe are following a continuous function but with random changes. By using the same algorithm for deciding whether there is a change (and thus a notification) at the members, we make sure that value of p_{change} in Equation (4.5) is constant across experiments. The experiments had different durations and the number of member notifications varied between them. The average number of notifications sent per member per experiment was 35, the minimum was 8 and the maximum was 103 notifications.

4.6.3.2 Entity precision

The second technique clients can use in order to optimize the number of notifications they receive from the EM is the Entity Precision (Section 4.5.2). If a client is not interested in the full precision of the sensor data, it can request the EM to reduce the precision. Here again, reducing the precision also typically reduces the number of client notifications, since consequent member notifications values might be the same when their precision is reduced, not resulting in a new client notification.

As calculated in Equation (4.6), the ratio $r_{precision}$ between the number of client notifications to member notifications is independent from the group size. In fact $r_{precision}$ depends only on p_{same} which is the probability that two consequent values from a single member will be the same if their precision is reduced. We have validated this behavior experimentally and show the results in Figure 4.17. In these experiments we used the Entity Operation `lst` which has no effect on $r_{precision}$ in order to be able to focus on the effect of reducing the precision on $r_{precision}$.

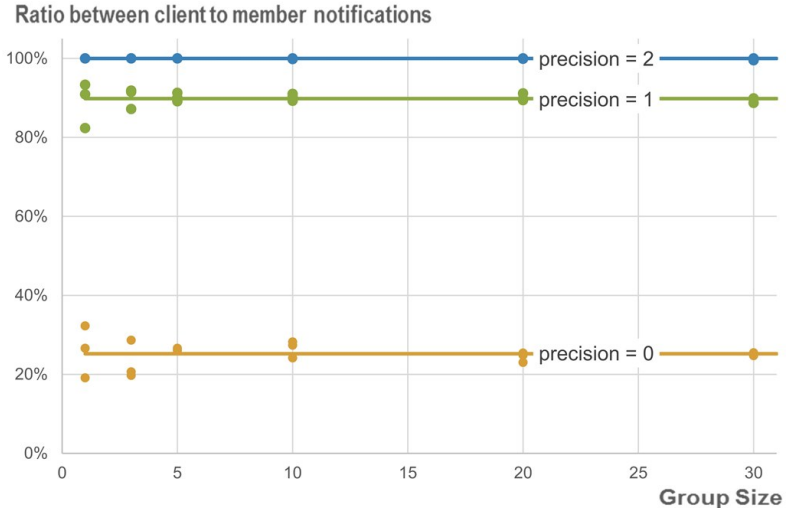


Figure 4.17: Reducing the precision of the entity can reduce the number of client notifications. The reduction ratio is independent from the group size.

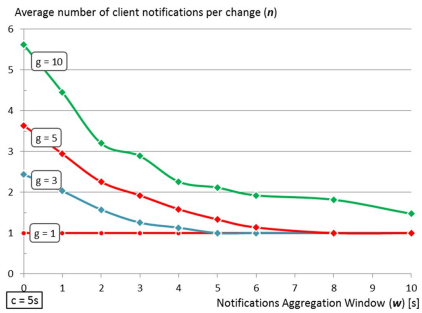
The figure shows the results of three sets of experiments for three different precisions: two, one and zero. As members, we have used software sensors in order to be able to change the precision of the resources they represent and have the precision set to two digits. Thus, when the EM used the same precision with the observing clients, all notifications were forwarded and hence $r_{precision} = 100\%$. When the client requested the reduction of the precision by one digit (i.e., $precision = 1$) and by two digits ($precision = 0$) $r_{precision}$ was reduced to 89.8% and 25.2%

respectively. The exact values of $r_{precision}$ depend on the pattern on which the data change. This is second reason why we used the software sensors and the same function for changing their values across all experiments (similar to Section 4.6.3.1). For very small group sizes (three members or fewer) the deviation between the results of different experiments is noticeable in the graph. However, as the groups get larger, the deviation gets smaller since the total number of received notifications from all members gets larger resulting in better accuracy of the experiments. The experiments had different durations and the number of notifications sent by the members varied between them. The average number of notifications sent per member per experiment was 81, the minimum was 9 and the maximum was 275 notifications.

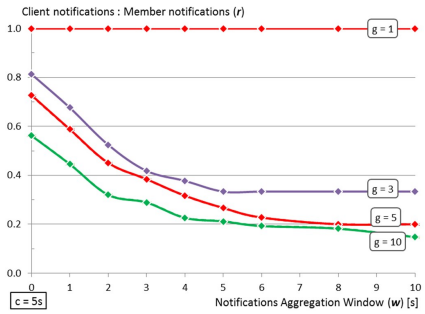
4.6.3.3 Notifications aggregation window

In addition to the Entity Operation and Entity Precision that are explained in the previous subsections, clients can specify a Notifications Aggregation Window w (see Section 4.5.3). This window tells the EM the number of seconds up to which the EM can delay notifications before sending them to the client. By allowing the EM to delay some notifications, it becomes possible for the EM to combine several member notifications into a single client notification and thus decreasing the number of notifications the client receives. The reduction in the number of client notifications depends on the timing of the member notifications. If multiple sensors monitor a single environment, then it is likely that a change in this environment will be registered by those sensors in a relatively short period. If these sensors are the members of an observed entity, then the notifications that the individual members send to the EM are not independent from each other; they are correlated. When using an optimal Notifications Aggregation Window, the EM will capture all of those member notifications into a single client notification.

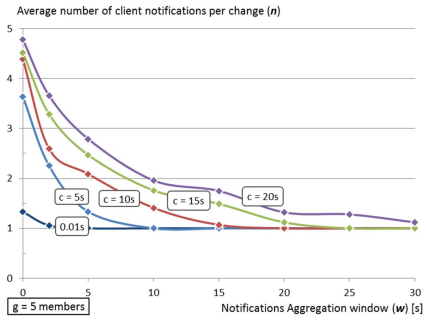
In order to investigate the effect of introducing the Notifications Aggregation Window on the number of client notifications, we have conducted several experiments using members running in Cooja network simulator. The use of Cooja allows us to accurately control the timing of notifications sent by the individual members to the EM so that we are able to create tests in which the notifications from different members arrive at the EM within a certain period. We call this period the *change window* c. In all experiments in this section we introduce an environmental change that can be monitored by all members. Each member notifies the EM about the change after a random time between zero and c. This way we simulate the correlation between the notifications from different members. We have conducted several experiments to explore the effect of the three variables: Notifications Aggregation Window (w), change window (c) and group size (g) on the number of client notifications (n) and the ratio between client and member notifications (r). In each set of experiments we kept two variables constant and



(a) n vs w; $c = 5s$.



(b) r vs w; $c = 5s$.

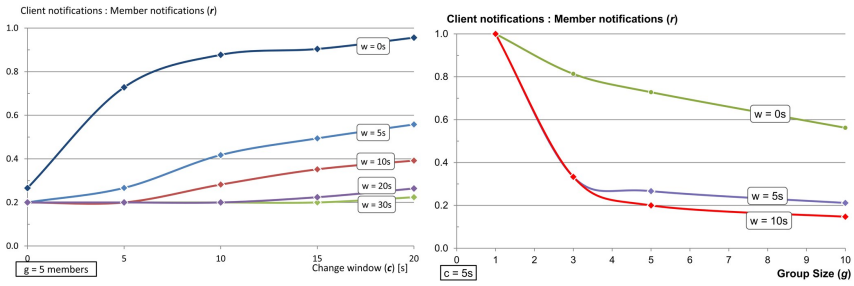


(c) n vs w; $g = 5$.

Figure 4.18: The effect of the Notifications Aggregation Window size (w) on the number of client notifications (n) and the ratio between client and member notifications (r). (c: change window, g: group size.)

changed the third variable and measured its effect. We let each experiment run for a duration long enough to allow at least 50 notifications from each member.

Effects of the size of Notifications Aggregation Window. In the first set of experiments (Figure 4.18a) we used a fixed change window ($c = 5s$) and changed the Notifications Aggregation Window ($w = 0 \rightarrow 10s$) and measured the average number of client notifications (n) for different group sizes ($g = 1, 3, 5, 10$). As expected, the graph shows how an increase in w results in an exponential decrease in n, as multiple member notifications are aggregated into a single client notification than smaller groups. Nevertheless, when w becomes sufficiently large, a single notification is sent to the client, containing all member notifications. Obviously, larger groups result in more notifications than smaller groups. However, if we normalize the number of notifications by the group size, we get the ratio



(a) n vs c; $g = 5$. (b) n vs g; $c = 5s$.

Figure 4.19: The effects of the change window (c) and the group size (g) on the ratio between client and member notifications (r) for various Notifications Aggregation Window sizes (w).

between client and member notifications r regardless of the group size. Figure 4.18b shows how larger groups achieve a smaller r (better reduction of the number of client notifications). The reason for this is that with bigger groups the number of client notifications that arrives at the EM in a single w gets larger. The functions converge to $1/g$ which means that when w is big enough the EM combines all correlated notifications from the individual members in one client notification.

In the second set of experiments (Figure 4.18c) we used a fixed group size ($g = 5$) and changed the Notifications Aggregation Window ($w = 0 \rightarrow 10s$) and measured the average number of client notifications (n) for different change window sizes ($c = 0.01s, 5s$, $10s$, $15s, 20s$). Here we see that larger change windows result in more client notifications being sent. This is due to the fact that with the increase of c it becomes less likely that member notifications will fit in the same Notifications Aggregation Window. When w becomes sufficiently large, the EM sends a single notification to the client.

Effects of the change window size. In the third set of experiments (Figure 4.19a) we used a fixed group size ($g = 5$) and changed the change window ($c = 0.01 \rightarrow 20s$) and measured the ratio between client and member notifications (r) for different Notifications Aggregation Window sizes ($w = 0s, 5s, 10s, 20s, 30s$). One can clearly see here that when c is very small, all member notifications are aggregated in one client notification and thus $r = 1/g = 0.2$ since $g = 5$. Please note that c can not get a lot smaller than 0.01s since this

is close the frame transmission time. If two members try to transmit with a very high degree of synchronization, collision between them becomes very likely and causes the MAC protocol to back-off before trying to retransmit later.

Effects of the group size. In the fourth set of experiments (Figure 4.19b) we used a fixed change window size ($c = 5s$) and changed the group size ($g = 1 \rightarrow 10\ members$) and measured the ratio between client and member notifications (r) for different Notifications Aggregation Window sizes ($w = 0s, 5s, 10s$). Here again the reverse exponential relationship between r and g is clearly visible. An interesting aspect that this figure reviles is that even when the $w = 0$ the EM manages to reduce client notifications when they are correlated as in these experiments. This is implementation dependent, as our implementation periodically checks if the EM has pending client notifications to send. When the member notifications are correlated, sometimes more than one notification arrive within this short period. When the group gets bigger, the chance that this happens get bigger as well.

4.6.3.4 Combination of properties

The previous subsections have shown that the EM provides three techniques that allow the client to reduce the number of notifications they get. Clients can use any of those techniques individually or may elect to combine any two or all three of them to achieve an even better overall optimization. To demonstrate this please consider Figure 4.20, where the results of four set of experiments are shown. First the client does not select any optimization technique and thus receives all member notifications as individual client notifications (i.e., $r = 1$). In the second experiment the client chooses to reduce the precision from 1 decimal digits to integer values and gets a reduction of about 10% of the number of the notifications regardless of the group size. In the third experiment the client uses the Entity Operation `avg` to get notified when the average value (with 1 digit precision) of all members changes. Finally, the client selects to combine both techniques (`eo=avg` and `precision=0`) and gets the benefits of both techniques combined.

4.7 Conclusions and outlook

In this paper we have tackled the problem of observing a group of CoAP resources by further exploring the potential of our unicast-based CoAP group communication approach. We have demonstrated that by relying on standard CoAP unicast messages, it becomes possible to observe CoAP groups effectively. We have also

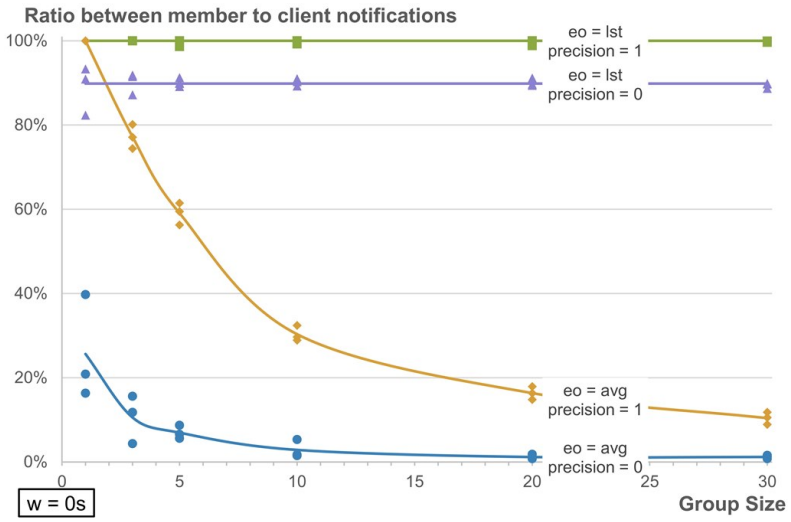


Figure 4.20: Combining entity properties achieves even better optimization.

shown that several techniques can be applied in order to further reduce the number of notifications sent to the observer. The amount of reduction in the number of notifications depends on the size of the group, the entity operation, and on the correlation between the values of different members. In our test examples we were able to reduce the number of notifications down to just 1% of the original number by reducing the data precision and using the Entity Operation `avg`. This approach fits within the current spirit of distributed processing or fog computing, where processing functionality is moved closer to the data sources, as bandwidth toward the Cloud is not infinite, might be costly or might cause to high latencies.

An advantage of our approach is that it requires no changes to sensors, i.e., one can add it afterward to any deployed CoAP network. Also it requires no changes to the clients, since groups behave as any other CoAP source. In the future, we plan to evaluate our approach in more detail. We will test the principles by using sensor data collected in real-life settings, both uncorrelated and correlated data.

An open challenge in our approach is that machines need to know more details about the data they are getting from the various sources in order to be able to process this data correctly. This includes the need for strict typing of data to be able to understand the contents of the resources. In addition to the CoRE link format, several initiatives that provide guidelines and standards in this regard are noteworthy. The *Internet Protocol for Smart Object (IPSO) Alliance* has published an *Application Framework* that recommends a classification of resources based on their functionality by defining a set of resource types [34]. Another example is the *Open Mobile Alliance* (OMA) *Lightweight M2M* (LWM2M), which is an open

industry standard that specifies a way to remotely manage a wide range of M2M devices and connected appliances in the IoT [35]. In the future, we will build upon such standards to further extend the CoAP++ framework to automatically interpret resource representations (content formats), data units, possible operations, etc. More work is needed to support automatic conversion between content formats and data units. Also standardization is needed to some extent, e.g., profiles of groups.

We like to further compare our approach with MQTT and see whether we can also apply QoS. In order to improve the performance of our solutions, we like to let the EM automatically adapt the Notifications Aggregation Window size based on the history of the distribution of notifications. Finally, further improvements might be possible by intervening at the routing and MAC levels in the LLN.

Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 258885 (SPITFIRE project), from the iMinds ICON projects O'CareCloudS and a VLIR PhD scholarship to Isam Ishaq.

References

[1] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. V. d. Abeele, E. D. Poorter, I. Moerman, and P. Demeester. *IETF Standardization in the Field of the Internet of Things (IoT): A Survey*. Journal of Sensor and Actuator Networks, 2(2):235–287, 2013.

[2] *Constrained RESTful Environments (core)*, 2014. [Online; accessed 19 March 2014]. Available from: http://datatracker.ietf.org/wg/core/.

[3] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252, IETF, June 2014. Available from: http://tools.ietf.org/html/rfc7252.

[4] A. Rahman and E. Dijk. *Group Communication for the Constrained Application Protocol (CoAP)*. RFC 7390, IETF, October 2014. Available from: http://www.ietf.org/rfc/rfc7390.txt.

[5] K. Hartke. *Observing Resources in CoAP*. Internet-Draft draft-ietf-core-observe-16, IETF, December 2014. Available from: http://tools.ietf.org/html/draft-ietf-core-observe.

[6] I. Ishaq, J. Hoebeke, F. Van den Abeele, J. Rossey, I. Moerman, and P. Demeester. *Flexible Unicast-Based Group Communication for CoAP-Enabled Devices*. Sensors, 14(6):9833–9877, 2014.

[7] I. Ishaq, J. Hoebeke, and V. den Abeele. *CoAP Entities*. Internet-Draft draft-ishaq-core-entities-00, IETF, June 2013. Available from: http://tools.ietf.org/html/draft-ishaq-core-entities.

[8] *Cisco Fog Computing with IOx*, 2015. [Online; accessed 5 March 2015]. Available from: http://www.cisco.com/web/solutions/trends/iot/cisco-fog-computing-with-iox.pdf.

[9] F. Van den Abeele, J. Hoebeke, G. K. Teklemariam, I. Moerman, and P. Demeester. *Sensor Function Virtualization to Support Distributed Intelligence in the Internet of Things*. Wireless Personal Communications, 81(4):1415–1436, 2015.

[10] M. Jung and W. Kastner. *Efficient group communication based on Web services for reliable control in wireless automation*. In Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE, pages 5716–5722. IEEE, 2013.

[11] M. Antonini, S. Cirani, G. Ferrari, P. Medagliani, M. Picone, and L. Veltri. *Lightweight multicast forwarding for service discovery in low-power IoT networks*. In Software, Telecommunications and Computer Networks (SoftCOM), 2014 22nd International Conference on, pages 133–138. IEEE, 2014.

[12] Z. Shelby. *Constrained RESTful Environments (CoRE) Link Format*. RFC 6690, IETF, August 2012. Available from: http://www.ietf.org/rfc/rfc6690.txt.

[13] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. *Fog computing and its role in the internet of things*. In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pages 13–16. ACM, 2012.

[14] H. Hasemann, O. Kleine, A. Kröller, M. Leggieri, and D. Pfisterer. *Annotating real-world objects using semantic entities*. In Wireless Sensor Networks, pages 67–82. Springer, 2013.

[15] H. Hasemann, A. Kroeller, and M. Page. *RDF Provisioning for the Internet of Things*, 2012. [Online; accessed 6 June 2015]. Available from: http://iot-conference.org/iot2012/uploadfiles/file/ppt/F1B-1%20RDF%20Provisioning%20for%20the%20Internet%20of%20Things.pdf.

[16] C.-D. Hou, D. Li, J.-F. Qiu, H.-L. Shi, and L. Cui. *SeaHttp: A Resource-Oriented Protocol to Extend REST Style for Web of Things*. Journal of Computer Science and Technology, 29(2):205–215, 2014.

[17] G. Bovet, G. Briard, and J. Hennebert. *A Scalable Cloud Storage for Sensor Networks*. In Proceedings of the 5th International Workshop on Web of Things, pages 22–27. ACM, 2014.

[18] *Message Queue Telemetry Transport MQTT*, 2015. [Online; accessed 2 March 2015]. Available from: http://mqtt.org/.

[19] W. Colitti, K. Steenhaut, and N. De Caro. *Integrating wireless sensor networks with the web*. In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011), Chicago, IL, USA, April 2011.

[20] A. Dunkels et al. *Efficient application integration in IP-based sensor networks*. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pages 43–48. ACM, 2009.

[21] M. Nottingham. *Web Linking*. RFC 5988, IETF, October 2010. Available from: http://www.ietf.org/rfc/rfc5988.txt.

[22] S. Loreto, P. Saint-Andre, S. Salsano, and G. Wilkins. *Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP*. RFC 6202, IETF, April 2011. Available from: http://www.ietf.org/rfc/rfc6202.txt.

[23] M. Buettner, G. V. Yee, E. Anderson, and R. Han. *X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks*. In Proceedings of the 4th international conference on Embedded networked sensor systems, pages 307–320. ACM, 2006.

[24] Z. Shelby and C. Bormann. *CoRE Resource Directory*. Internet-Draft draft-ietf-core-resource-directory-02, IETF Secretariat, November 2014. Available from: http://tools.ietf.org/html/draft-ietf-core-resource-directory.

[25] C. Jennings, Z. Shelby, and J. Arkko. *Media Types for Sensor Markup Language (SENML)*. Internet-Draft draft-jennings-senml-10, IETF, October 2012. Available from: http://tools.ietf.org/html/jennings-senml.

[26] F. Van den Abeele, J. Hoebeke, I. Ishaq, G. K. Teklemariam, J. Rossey, I. Moerman, and P. Demeester. *Building embedded applications via REST services for the Internet of Things*. In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, page 82. ACM, 2013.

[27] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester. *Facilitating sensor deployment, discovery and resource access using embedded web services*. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, pages 717–724. IEEE, 2012.

[28] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. *The Click modular router*. ACM Transactions on Computer Systems (TOCS), 18(3):263–297, 2000.

[29] *Zolertia Z1 Platform*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.zolertia.com/ti.

[30] *Rm090 — RMONI wireless nv*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.rmoni.com/en/products/hardware/rm090.

[31] *Contiki: The Open Source Operating System for the Internet of Things*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.contiki-os.org/.

[32] *Erbium (Er) REST Engine and CoAP Implementation for Contiki*, 2015. [Online; accessed 27 February 2015]. Available from: http://people.inf.ethz.ch/mkovatsc/erbium.php.

[33] G. K. Teklemariam, J. Hoebeke, I. Moerman, and P. Demeester. *Facilitating the creation of IoT applications through conditional observations in CoAP*. EURASIP Journal on Wireless Communications and Networking, 2013(1):1–19, 2013.

[34] *The IPSO Application Framework*, 2012. [Online; accessed 5 March 2015]. Available from: http://www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf.

[35] *OMA Lightweight M2M (LWM2M) protocol*, 2012. [Online; accessed 5 March 2015]. Available from: http://openmobilealliance.hs-sites.com/lightweight-m2m-specification-from-oma.

5

Experimental Evaluation of Unicast and Multicast CoAP Group Communication

"Liberty is the possibility of doubting, the possibility of making a mistake, the possibility of searching and experimenting, the possibility of saying No to any authority–literary, artistic, philosophic, religious, social and even political."

– Ignazio Silone (1900 - 1978)

Chapter 3 has introduced the multicast-based CoAP group communication solution and has provided a comparison to unicast-based solutions through simulations. However, simulations are often not representative for the real world, where constrained devices need to coexist with other devices and share the same wireless frequency bands. Therefore, in this chapter, we experimentally evaluate the two proposed solutions for CoAP group communication using large-scale testbeds. At the time of writing this Chapter CoAP was published as an RFC and the CoAP group communication was published as an experimental RFC. Also the used Contiki version included multicast support in the distribution.

⋆ ⋆ ⋆

I. Ishaq, J. Hoebeke, I. Moerman, and P. Demeester.

Submitted to the Journal of Network and Computer Applications, August 2015.

Abstract The Internet of Things (IoT) is expanding rapidly to new domains in which embedded devices play a key role and gradually outnumber traditionally connected devices. These devices are often constrained in their resources and are thus unable to run standard Internet protocols. The Constrained Application Protocol (CoAP) is a new alternative standard protocol that implements the same principals as the Hypertext Transfer Protocol (HTTP), but is tailored towards constrained devices. In many IoT application domains devices need to be addressed in groups in addition to be addressable individually. Two main approaches are currently being proposed in the IoT community for CoAP based group communication. The main difference between the two approaches lies in the underlying communication type: multicast versus unicast. In this article, we experimentally evaluate those two approaches using two wireless sensor testbeds and under different test conditions. We highlight pros and cons of each of them and show how these approaches can be combined to better suit certain use case requirements. Additionally we provide a solution for multicast-based group membership management using CoAP.

5.1 Introduction

It is not a secret that the Internet is expanding rapidly in all directions. On the one hand, within traditionally connected domains the number of devices and variety in device types are gradually increasing. On the other hand, traditionally unconnected domains are discovering the benefits of the Internet and are joining it continuously. The resulting Internet is now referred to as the IoT to stress the fact that it is connecting all sorts of things – not just people, computers and smart phones as one is used to nowadays.

It is expected that the IoT will soon contain more embedded devices than the traditionally connected devices. These devices are often constrained in their resources since they are optimized for low cost and/or battery operation. Most commonly these devices have constraints in terms of the available amount of Random Access Memory (RAM), Read Only Memory (ROM), and Central Processing Unit (CPU) power. They are often required to consume very little power, since they are often battery powered or rely on energy harvesting. Ideally these devices should run for years without the need for costly battery replacements. Not only the IoT devices have constraints, but also the networks in which they operate have their own constraints. These networks typically have lower bandwidths and higher error rates than common Internet networks. These networks are usually referred to as Low-power and Lossy Networks (LLNs).

As a result of device and LLN constraints the use of standard Internet proto-

cols becomes often unfeasible, since these protocols were not designed to accommodate for those constraints. In the past few years, there were many efforts to enable the extension of the Internet technologies to constrained devices. Most of these efforts were focusing on the networking layer, e.g., IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [1] and IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [2]. More recent, work has started to allow integration of constrained devices in the Internet at service level. The Internet Engineering Task Force (IETF) Constrained RESTful Environments (CoRE) working group has realized the Representational State Transfer (REST) architecture in a suitable form for the most constrained nodes and networks. As a result the CoAP was introduced, a specialized RESTful web transfer protocol for use with constrained networks and nodes. CoAP realizes a subset of REST that is common with the HTTP, but is optimized for Machine-to-Machine (M2M) applications such as smart energy and building automation [3]. With the introduction of CoAP a complete open standard networking stack suitable for constrained devices is now available [4]. Constrained devices are turned into embedded web servers that make their resources accessible via the CoAP protocol.

In many IoT application domains, the devices need to be addressed in groups in addition to being addressable individually. For example, in a smart building, it should be possible to manually or automatically control the opening of all window shutters at a certain side of the building based on the position of the sun. This operation of the shutters as a group should not hinder the ability of the user to control a single shutter individually or to control only those shutters that belong to her room. Two main approaches are currently being proposed in the IoT community for CoAP based group communication. The main difference between these two approaches lies in the underlying communication type: multicast versus unicast. The experimental standard for CoAP group communication [5] relies on Internet Protocol version 6 (IPv6) multicasts while our approach proposed in [6] relies on IPv6 unicast messages. In this article we experimentally evaluate those two approaches using two wireless sensor testbeds representing different testing environments. The first testbed is located in a large, shielded room, which allows testing under controlled environments. The second testbed is located inside an operational office building and thus allows testing under normal operation conditions. We highlight the pros and cons of each of group communication approaches and show how they can be combined to better suit certain use case requirements. To our knowledge, no other work has done such evaluations at large scale.

In addition to the experimental evaluation we present our extension that allows the use of CoAP for the management of the multicast group membership. Using this simple extension, it becomes possible to ask CoAP enabled devices to join and leave multicast groups without the need for manual intervention or a dedicated management protocol. This is an essential feature in dynamic settings,

where group membership is changing frequently.

The remainder of this article is organized as follows. In Section 5.2 we provide the motivation for this work and briefly show the need for group communication in various application domains of the IoT in general and elaborate a bit more on the building automation use case. We then provide a brief introduction to CoAP and introduce the different approaches for CoAP group communication in Section 5.3. In Section 5.4 we introduce our CoAP group communication implementation and evaluate it in Section 5.5. We then discuss related work in Section 5.6 before concluding this paper in Section 5.7.

5.2 Motivation and requirements

In this section we provide the motivation for this work. We start by briefly showing the need for group communication in various IoT application domains in general. Then we elaborate on those needs for Building Automation Systems (BASs), which we take as a use case in the remainder of this article. Finally we derive the requirements for CoAP group communication based on those needs.

5.2.1 Need for group communication in the IoT

The basic CoAP interaction model is based on one-to-one communication, i.e. a client is exchanging messages with a server. The basic interaction model covers a wide range of the interaction patterns that are found in various fields of the IoT. However, in many use cases, a one-to-many communication pattern is convenient or even essential for various reasons. Sometimes the data collected from sensors might not be sufficient to get the complete information about the monitored object. In other cases it is desired to get the information from more than one source to increase accuracy or reliability. Sometimes, data from many sources needs to be collected and processed before it can be used. Likewise, it is often needed to control more than one actuator at once to make the complete object act as desired. This need to communicate with groups of objects is obvious in many IoT scenarios. For example, in transport logistics, it might be necessary to update all containers from a single source or a single destination at once. In a smart farm, it is handy to be able to activate irrigation for all trees from a certain type and age at once or to query the last time all pregnant sheep went to drink water. Similar needs can be found in a lot of the IoT application fields.

It is often possible to realize many of these interaction patterns by using only basic one-to-one COAP communication and letting the client interact with each sensor or actuator individually. This typically leads to more complex applications, since the application needs to have the intelligence about the various individual resources and how to process the collected data. This might also lead to ineffi-

cient - and possibly costly - network usage when remote clients are involved as all individual resource values have to travel to the remote client individually.

By allowing clients to interact with many sensors or actuators at once with a single request, intelligence and possibly processing can be pushed away from the client towards the data sources. By doing so, more efficient network usage can be achieved which often results in faster response times.

The need for group communication is very well recognized in the IETF as it is part of the charter of the IETF CoRE Working Group [7]. The relevance of multicast communication for wireless networks for BASs has been identified in [8]

5.2.2 Use case: Building Automation Systems

In this paper we focus on group communication in BASs, which is in line with the experiments we conduct on the testbeds in Section 5.5. In this subsection, we first discuss the different types of connected devices in such systems and how such systems could benefit from grouping patterns.

5.2.2.1 Types of connected devices

Building automation and management is one of the domains that has caught a lot of attention as promising domain for IoT application. In the beginning various vendors developed their own proprietary communication protocols to control their systems (light, energy, etc.). This made communication between systems from different vendors difficult and sometimes impossible. With the standardization of CoAP and the other underlying open standard networking stack, it is now possible to communicate between different systems and devices from various vendors and with other connected IT devices typically present in buildings. Since this networking stack can run on the most constrained devices, it is now possible to interconnect different types of devices (constrained and not constrained) in one big home network. This opens the doors for application developers to develop applications that use features from different systems and devices to enhance the user experience.

Virtually all BASs need – or at least can benefit from – some sort of group communication features. Following is a simple discussion of those needs starting from the types of devices that will be present in future BASs, which will help us identify a set of requirements for group communication in the next subsection.

Heating, Ventilating, and Air Conditioning (HVAC) systems are a very important application domain for the IoT, since these systems typically consume a lot of energy. On the one hand, if these systems are not managed properly, they can waste a considerable amount of energy. On the other hand, these systems allow for a considerable amount of automation and can be effectively managed to enhance

the user experience and save energy at the same time. These systems typically need to control individual components at an individual basis, but they also often require to control groups of units at the same time. For example, in an office building it might be necessary to shut down all heating units in a large room when the room is put into sleep mode. Another example is to log the average temperature of the room by querying all temperature sensors in that room.

Lighting Systems is another aspect that can benefit a lot from group communication. Any single light should be controllable individually but the same light should be controllable as a member of a group. Here groups can represent the location of the lights (e.g., room 3.17), the function of the light (e.g., spot light), a combination of both (e.g., spot lights in room 3.17) or any other grouping criteria. The same applies for querying the status of the lights.

Building Access Control Systems allow among others the reprogramming of door locks to grant access to users based on various conditions. One example that benefits from group communication is to allow access to all participants of a meeting to the meeting room. This might involve not just the reprogramming of the locks at the meeting room doors, but also all locks on the hallway doors from the building entrance up to the room itself.

Energy Management involves the ability to monitor and being able to interact with other BASs. For example, it is important to be able to put rooms in sleep mode when the calendar system indicates that the rooms will not be used for longer periods. In such cases the energy management system could request that all heating units are being put to a lower temperature and the lights reduced to only emergency lighting.

5.2.2.2 Grouping

Based on the examples listed in the previous subsection one can see that there are many ways to group resources. Following are some possible types or motivations for resource grouping in BASs.

Physical grouping creates groups that reflect the physical structure of building (i.e. desk, part of room, room, wing, floor, building, and so on). In such a hierarchical structuring, resources should be able to join more than one group or/and groups should be able to have other groups as members. Groups might contain only members that are homogeneous (i.e. same type of device) or heterogeneous.

Logical grouping creates groups of similar sensors or of sensors that offer the same functionality, e.g., all light intensity sensors. Typically these are homogeneous groups, i.e. they offer a uniform API to access their resources.

Application grouping creates groups from an application point of view. These group can be heterogeneous, i.e. a single group can be created for reading the power consumption of many heterogeneous devices or to obtain diagnostic information about them.

Administrative grouping involves the creation of groups according to roles, rights or policies. For instance, in a building context, there are many possible stakeholders such as owners, tenants, visitors, administrators, etc. Different groups can be created from a role-based point of view and access to these groups is then linked to the role of the users. Similarly, groups can be created that are only available during certain times of the day. Through administrative grouping, the management of and access to batches of devices can be realized in a flexible way.

5.2.3 Requirements

Within this context, we have defined the requirements and goals for CoAP group communication, which are described in detail in [6]. We have motivated their importance in the context of IoT applications, constrained devices and LLNs. For completeness of this article, we summarize these requirements as follows:

1. *Flexibility*: The group communication solution should be very flexible to accommodate the differences between devices and device types. In particular the solutions should be flexible enough to support homogeneous (same URI, media-type, etc.) and heterogeneous groups and support group members that are not part of the same network.
2. *Light-Weight (footprint)*: the group communication solution should have limited footprint on constrained devices. Furthermore the solution should scale with the number of groups a certain member can be part of.
3. *Use of Standards*: to allow the creation of groups across a variety of members from different vendors and domains, it is mandatory to use standard protocols that are widely supported. The focus of our work is on using CoAP as an application layer standard protocol. It is essential to limit the use of optional CoAP extensions in order not to exclude potential CoAP servers of becoming group members due to missing extensions.
4. *Performance*: Any CoAP group communication solution should have little overhead and be efficient in the use of resources of the nodes and the LLN and should not hinder the use of caches.

5. *Validation and Error Handling*: It should be possible to validate the group in order to make sure that the group works as intended. The group should have mechanisms for reporting and handling error conditions such as node or route failures.

6. *Reliability*: sometimes it is not essential to get reliable replies from all group members (e.g., it might be enough to get the temperature measurements from just one of the many temperature sensors in a room). However in many other cases, it can be important to have reliable communication with all group members. For example, one would expect that all lights in the room would go on when one flips on the light switch.

7. *Ease of Group Manipulation*: the needs of the user might change with time and thus group membership might also change. In dynamic environments the changes might be frequent. It is important to be able to handle such changes easily. One should avoid node reconfigurations, as this might be a tedious task. Also it should be possible for nodes to be part of different groups at the same time or at different times.

8. *Expressiveness/Control*: It is desired to have support for processing of individual group member results and replying to the requester with aggregated results. For example it should be possible to query a certain subset of the members and compute the average, or reliably update all members.

9. *Security*: By exposing sensors and actuators to the Internet, security becomes a major concern. In some scenarios having end-to-end security is a strict requirement. Communicating with a group of resources is no exception. In fact it is even more sensitive than communicating with an individual resource, since compromising the group means compromising all the individual members.

5.3 CoAP group communication

There are two main approaches to allow CoAP to be used for the interaction with groups. The first approach relies on IPv6 multicasts and the second relies on IPv6 unicasts. These two communication methods have different characteristics that heavily influence the respective approaches that use them. In this section we discuss the different approaches to realize CoAP based group communication, after briefly introducing the key features of the CoAP protocol.

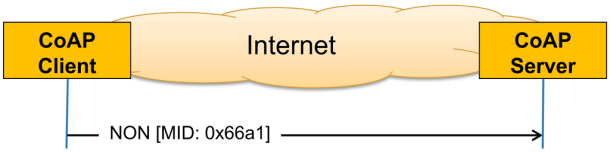


Figure 5.1: Example of CoAP Non-confirmable Message (NON) exchange. A Message ID (MID) is needed in the header for duplicate detection.

5.3.1 The Constrained Application Protocol (CoAP)

Recent research on embedded web services is laying the ground for a better integration of sensor resources into the service web. Since the dominating web protocol HTTP is too complex, the IETF CoRE working group [3] has designed a simpler web protocol – CoAP and defined it in Request For Comment (RFC) 7252 [9]. CoAP uses the same RESTful principles as HTTP, but it is much lighter so that it can run on constrained devices [10, 11]. To achieve this, CoAP has a much lower header overhead and parsing complexity than HTTP. It uses a 4-bytes base binary header that may be followed by compact binary options and payload.

The CoAP interaction model is similar to the client/server model of HTTP. A client can send a CoAP request, requesting an action (specified by a method code) on a resource (identified by a URI) on a server. The CoAP server processes the request and sends back a response containing a response code and payload.

A message that does not require reliable transmission can be sent as a Non-confirmable Message (NON). This type of messages is not acknowledged, but has a Message ID for duplicate detection (see Figure 5.1) Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport such as User Datagram Protocol (UDP) and thus a NON might get lost without the client and the server noticing it. Multicast CoAP requests are sent using NONs.

Since UDP does not provide reliable communication, optional reliability is supported within CoAP itself. This is done by using Confirmable Messages (CONs) to implement simple stop-and-wait retransmissions with exponential back-off. Figure 5.2 shows an example for a confirmable message exchange. The MID must be the same in order to match ACKs with CONs and for duplicate detection. The Token must be the same in order to match replies with requests. If the client does not receive an ACK for its CON within an initial back-off time, it retransmits the same CON again. By default the initial back-off is set to a random time between 2s and 3s. This means that if a reply to a confirmable packet is not received within the initial back-off time, the CoAP sender will double the initial back-off time and retransmit the packet. If a reply to the first retransmission is not received, CoAP will again double the back-off time and retry the transmission until MAX_RETRANSMIT (by default 4) is reached. If no reply is received after expiry of the back-off

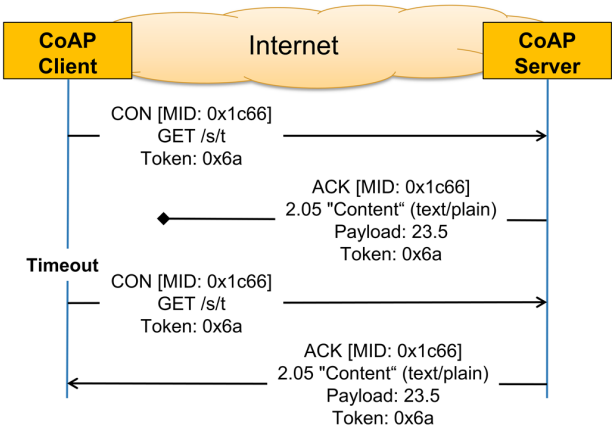


Figure 5.2: Example of CoAP Confirmable Message (CON) exchange. If the client does not receive an ACK for its CON within a certain time, it retransmits the same CON again until it gets acknowledged or until the client runs out of retransmission attempts.

time of the last retransmission, the client will be notified about the error condition.

The base CoAP protocol can be enhanced with multiple optional extensions. While all CoAP clients and servers should be able to communicate using the base CoAP protocol, the implementation of the other CoAP extensions is optional. By doing so, it becomes possible to keep the base protocol as small and as efficient as possible to fit on even the most constrained devices and at the same time to allow the protocol to be extended to suite more advanced communication needs. One example of such an extension is the observe option which allows clients to register their interest in receiving notifications whenever a resource at the server changes its value. This eliminates the need for clients to continuously pull a resource to see whether it has changed. Finally, CoAP supports optional security by using Datagram Transport Layer Security (DTLS) [12].

5.3.2 Multicast group communication

The IETF CoRE working group has recognized the need to support a non-reliable multicast message to be sent to a group of devices to manipulate a resource on all the devices in the group. Therefore, they have developed the "Group Communication for CoAP - RFC 7390" [5], which provides guidance for how the CoAP protocol should be used in a group communication context. Group Communication refers to sending a single CoAP non-confirmable message to all members of a specific group by utilizing UDP/IP multicast for the requests, and unicast UDP/IP for the responses (if any). This implies that all the group members (the destination nodes) receive the exact same message.

The use of multicast is efficient in sending the requests, but does not affect the number of responses sent by the members since these are sent as unicasts. However, the use of multicasts has its limitations and challenges:

- The most prominent limitation is the **lack of reliability**, which makes it not suitable for all use cases.
- Another important limitation is the **cache-unfriendly** nature of multicasts preventing possible reduction of requests and replies by utilizing caches. Depending on the use case and network topology, the reduction of packets as a result of using a cache can be better than the reduction obtained from using multicasts.
- Also multicasts are **not useful when a single user action needs to trigger different sensor requests**, since one multicast request delivers the same message to all group members.
- **Secure communication with the group members is not possible**, since all communication based on this RFC operates in CoAP NoSec (No Security) mode.
- Finally, the use of multicast **requires multicast support in the network**. Typically IP multicast relies on topology maintenance mechanisms to discover and maintain routes to all subscribers of a multicast group. However, maintaining such topologies in LLNs may not be feasible given the available resources. As a result, special multicast protocols have been proposed for the use inside LLNs. For example, the "Multicast Protocol for Low power and Lossy Networks (MPL)" Internet draft uses the Trickle Multicast (TM) algorithm to manage message transmissions for both control and data-plane messages and avoids the need to construct or maintain any multicast forwarding topology [13]. An alternative is the Stateless Multicast RPL Forwarding (SMRF) algorithm, which according to [14] achieves significant delay and energy efficiency improvements at the cost of a small increase in packet loss. Regardless of the used multicast protocol, all nodes on the path between the sender and receivers must be extended to support the protocol.

5.3.3 Unicast group communication

In [6] we have introduced our alternative unicast-based group communication solution for CoAP-enabled devices. Our aim was to create an intermediate level of aggregation to be able to easily manipulate a group of resources across multiple smart objects. Such a group of resources is called an *entity* and the resources themselves are called the entity *members*. An entity can be created, used or manipulated through a single CoAP request.

We call the component that manages the entities, the *Entity Manager* (EM). This component, which can reside, e.g., on the Border Gateway (GW) of the LLN, is responsible for maintaining entities that are created from groups of resources residing on CoAP servers (i.e. sensors and actuators) inside the LLN. Clients on the Internet can interact with an EM to create new entities and/or customize how these entities should behave. The EM acts as a proxy between the client and the constrained devices. Client requests are sent to the EM, which analyzes and verifies the requests and then issues the appropriate requests to the constrained devices using CoAP. Once the EM receives responses from the constrained devices, it combines them according to the needs of the client and sends back an aggregated response to the client.

When a client tries to create a new entity consisting of a group of resources inside LLNs, the EM performs a sanity check on the request in order to make sure that the resulting entity would make sense. We call this sanity check the *Entity Validation*.

Furthermore, we have introduced in [6] the notion of profiles for the created entities. The use of entity profiles allows the client to specify in more detail how the entity should behave (e.g., if it should use confirmable or non-confirmable CoAP messages), and, through updating the profile, allows manipulation of this behavior. As such, we combined ease of creation, ease of usage and flexibility in behavior into a complete solution for interacting with CoAP resources from different objects inside a LLN. By building upon standardized concepts, the impact on the constrained devices was limited.

An important advantage of our approach is its reliability since it uses unicast messages that can make use of the CoAP built in reliability mechanism. Its main disadvantage is the increased communication overhead to achieve this reliability. For more details we refer the reader to [6].

5.3.4 Hybrid group communication

In the previous two subsections we have briefly described the two main approaches for CoAP group communication. The first approach uses CoAP NON over IPv6 multicast and the second one uses CoAP CON over IPv6 unicasts. We have shown that each approach has its strong and weak points. The multicast approach is generally fast and has little overhead, but is not flexible and not reliable and thus cannot be used when reliability is required. One the other hand the unicast approach is reliable and flexible, but is often slower and has more overhead.

In this paper, we extend our previous unicast-based solution with multicast support. We do so by maintaining the overall architecture of our approach. Clients still create Entities on the Entity Manager (EM) to represent the groups and have to query the CoAP group resource on the EM to access the group. However at

creation time clients can specify the communication type between the EM and the entity members (i.e. multicast or unicast). We have extended our existing solution so that all its features become valid for entities communicating by using multicast as they are valid for entities using unicasts. In this subsection we discuss the most important features and how they are extended to allow entities to use multicast communication.

5.3.4.1 Entity profile

Resource profiles can be used to express capabilities of a CoAP server and its resources [15]. In [6] we have extended the resource profiles concept with entity specific fields. By using Entity Profiles, it becomes possible to customize the default behavior of the entity. These entity profile fields can be provided by the client upon entity creation time. If no values are provided, the EM will use default values for the newly created entity. Clients can either use this default behavior or change any of those fields each time they use the entity by adding URI-queries to the request. Following is a list of the entity profile fields that are relevant to this current work.

- *Entity Resources (er)*: a list of the individual resources out of which the entity is composed.
- *Entity Message Type (emt)*: specifies the message type to be used for communication between EM and members. Possible values are *con* (confirmable) and *non* (non-confirmable). The default is *con*.
- *Entity Message Type (ect)*: specifies the communication type to be used for communication between EM and members. Possible values are *unicast* and *multicast*. The default is *unicast*. Whenever *ect* is set to multicast the EM sets *emt* to *non*, since CoAP supports only non-confirmable multicast requests. Please note that this is the only profile field that can be set only at entity creation time and cannot be set at usage time by using URI-queries. The reason for this is that changing the communication type implies changing the multicast group membership (see Section 5.3.4.2) which we consider as a configuration task that should not be done on the fly.
- *Entity Number of Replies (enr)*: specifies the number of replies that should be received from the members (from now on: member replies) before sending a reply to the client (from now on: client reply). This makes it possible not to wait for all members to reply. The default behavior is to wait until all replies have been received or have timed out. For example, the entity could consist of 5 members, but a reply from any 3 of them can be considered sufficient for the client and thus the EM does not need to wait until it receives

replies from all group members. This does not just increase the speed of the client reply, but also makes the group tolerant to failures. This feature is of particular importance, since the client could set `enr` to one. In this case the multicast entity behaves as an anycast group. This is an important feature, since RPL does not support anycast routing and thus this communication pattern was not yet available for constrained nodes.

- *Entity Operation (eo)*: The operations that can be performed on the results obtained from the members. The operation is used to combine replies received from all the members (or the number of replies specified by enr) into one reply to the client. If at the time of querying the entity the client does not specify which operation to use, the first operation listed in this field will be used. Currently the following Entity Operations are supported:
  - *List (lst)*: A list of replies received from the members, without any arithmetic processing. This is the default behavior if no entity operation was specified.
  - *Average (avg)*: The average value.
  - *Minimum (min)*: The minimum value.
  - *Maximum (max)*: The maximum value.
- *Delay between Requests (delay)*: specifies the delay that should be injected between the requests sent from the EM to the members. The default is 0 and thus the EM will send the requests as fast as it can.

5.3.4.2 Group membership management

Another important feature that our solution provides is the multicast group membership management. This means that whenever a multicast entity is created, the EM assigns an IPv6 multicast address to the entity and automatically requests from the individual members to join the multicast group. The EM verifies that the members have joined the group and informs the client about the assigned multicast address. Since the assigned multicast addresses can be globally routable, the client can communicate with the multicast group directly without contacting the EM. In order for members to be able to join the group, the members' constrained CoAP implementation has been extended with a new resource for group management. Details about the group membership management implementation will be provided in Section 5.4.1.

5.3.4.3 Summary

In summary, our proposed hybrid group communication solution for CoAP is an extension to our previously proposed unicast-based solution. By extending it with

multicast support, we achieved an increased flexibility of our solution. It is now up to the creator of the entity to decide whether communication reliability with the members is essential. When it is not essential and the user selects to create a multicast entity, the EM automatically configures the multicast group and lets the members join it. This minimizes manual and error prone group membership configurations. Since group membership management is done via unicast CoAP messages, it is reliable and the user can be informed about its results.

5.4 Implementation

The key in our group communication solution is the EM. We have implemented the EM functionality using the CoAP++ framework [16] and typically run it on the GW of the LLN. The CoAP++ framework and the EM implementation on top of it have been realized in *Click Router*, a C++ based modular framework that can be used to realize any network packet processing functionality [17].

As group members we have used Zolertia Z1s [18] and Rmoni RM090 [19] boards. On these boards we have run a development version[1] of the popular Contiki operating system [20]. This version of Contiki was the most recent development version when we started our experiments. It supports multicast and includes a stable implementation of CoAP, namely the Erbium CoAP server [21].

5.4.1 Multicast group management using CoAP

By default Contiki nodes join four multicast groups when multicast is enabled on them. All these groups have Link-Local Scope and thus the IPv6 Multicast Addresses of all of them start with ff02: (Table 5.1). The first three addresses are the same for all nodes, since these are generic addresses representing all nodes, all routers and all RPL enabled nodes on the local network segment. The last address is unique for each node and depends on the Node ID. This address is called the Solicited-node multicast address and is created by taking the last 24 bits of the nodes unicast address and appending it to the prefix ff02::1:ff00:0/104 [22]. The Solicited-node multicast address is used in neighbor discovery for obtaining the layer 2 link-layer addresses of other nodes [23].

In addition to the four default multicast groups that the nodes join automatically, the Contiki multicast implementation allows specifying other multicast groups to join as needed. However, the maximum number of groups to join is set at compile time. By default, a node can join only one group in addition to the groups listed in Table 5.1. This maximum number of groups that a node can join can be changed at compile time. For each additional multicast address, the node requires 18 bytes RAM. When using the nodes in a static setting, the multicast address(es)

[1] snapshot from the Contiki master branch on github on 2 July 2014

Table 5.1: When multicast is enabled on Contiki nodes, they automatically join four Link-Local groups.

Group IPv6 Address	Description
ff02::1	All nodes on the local network segment
ff02::2	All routers on the local network segment
ff02::1a	All RPL nodes on the local network segment
ff02::1:ff00:xx	Solicited-node multicast address. xx: Node ID

can be hard coded when the node is programmed. However, since many group communication scenarios require frequent group membership changes, it is essential to have a solution that allows changing a node's group membership on the fly without the need to reprogram it. We have provided such a solution using the same technology that we use to communicate with nodes, i.e. using CoAP. By using CoAP for this task as well, we eliminate the need for a dedicated group management protocol and can keep the solution lightweight.

In order to offer a RESTful interface for multicast group membership on the constrained devices, we extended the Erbium implementation on these devices with a new CoAP resource: `/mc`. This resource can be used to list the multicast addresses of the groups that a node has joined, to join a new group and to leave a group. Table 5.2 documents the interface to the group membership management resource.

Table 5.2: The group membership management resource `/mc` interface.

Method	URI-Query	Request Payload	Description
GET	–	–	Show membership
POST	act=add	IPv6 address (array of bytes)	Join group
POST	act=del	IPv6 address (array of bytes)	Leave group

The footprint of the multicast group membership resource is small. It requires only 1408 Bytes of ROM and 28 Bytes of RAM when compiled with gcc 4.6 for the RM090 nodes. In Table 5.3 we provide resource availability for the nodes that we used in our tests along with the resource requirements of the two available multicast implementations and our multicast membership management resource.

5.4.2 EM multicast extensions

We have extended our unicast-based solution to support communicating with the group members using IPv6 multicasts. We have also extended the existing features of our solution to support this new communication type. More specifically, we have defined a new profile field `ect` to specify the Entity Communication Type. This field can take two possible values, namely `ect=unicast` or

Table 5.3: Resource availability and requirements. The available resources on sample devices and the requirements for various system components.

	ROM (Bytes)	**RAM (Bytes)**
Class 1 device	$\approx 100k$	$\approx 10k$
Zolertia Z1	$92k$	$8k$
Rmoni RM090	$256k$	$16k$
TM (one user defined multicast address)	4176	1802
SMRF (one user defined multicast address)	1516	322
Additional multicast address	–	18
Multicast group management resource	1182	28

`ect=multicast`.

When a multicast entity is created, the EM assigns a new multicast group address to the entity and adds another field to the entity profile with the name `mcastURI`. This field specifies the URI to which a multicast address can be sent, when it is desired to communicate to the multicast group without going through the entity resource on the EM. This URI contains the IPv6 multicast address of the group and the resource name on the individual members e.g., `coap://[ff1e::89:1]/s/t`. Since the resource name has to be same across all group members, the EM adds this check to the validation process of multicast entities. In our implementation we used the following IPv6 multicast addresses `ff1e::89:xx` where the first 16 bits of the address (`ff1e`) indicate that the address is non-permanently-assigned ("transient" or "dynamically" assigned) multicast address with global scope and the `xx` is a 16-bit sequential number.

The EM then uses the group membership management resource on the nodes to let them join the newly assigned multicast address. The communication with entity management resource on the constrained devices is done via unicast CON messages to ensure reliability. The group creation is considered successful only if all required members successfully join the group. In any case the group creator is informed about the results of the multicast group join requests. Similarly, when a multicast group is deleted, all members are sent a unicast request to leave the multicast group and the deleter of the group is informed about the results.

5.5 Evaluation

When we evaluated our approach in [6], we have used our Demo box for the demonstration of the functionality [24] and the Cooja network simulator, which is part of the Contiki operating system, for the performance evaluations. The simulation environment enabled the initial evaluation of the performance of our solution for varying entity sizes and number of hops to the entity resources. However,

evaluation on larger real-life testbeds is required for validating the simulation experiments and for conducting experiments in denser and more realistic (e.g., Wi-Fi interference) environments. In this section, we will present an extensive evaluation of both multicast and unicast based solutions on two wireless sensor testbeds with different characteristics (Subsections 5.5.2 and 5.5.3). Before doing so, we first start by evaluating the functionality of the new extensions to our solution in (Subsection 5.5.1).

5.5.1 Functional evaluation

The two main features that we have added to our approach as originally presented in [6] are the support for multicast entities and the provisioning of more advanced features, such as the nesting of entities. In this subsection we functionally evaluate these two extensions.

5.5.1.1 Multicast entities

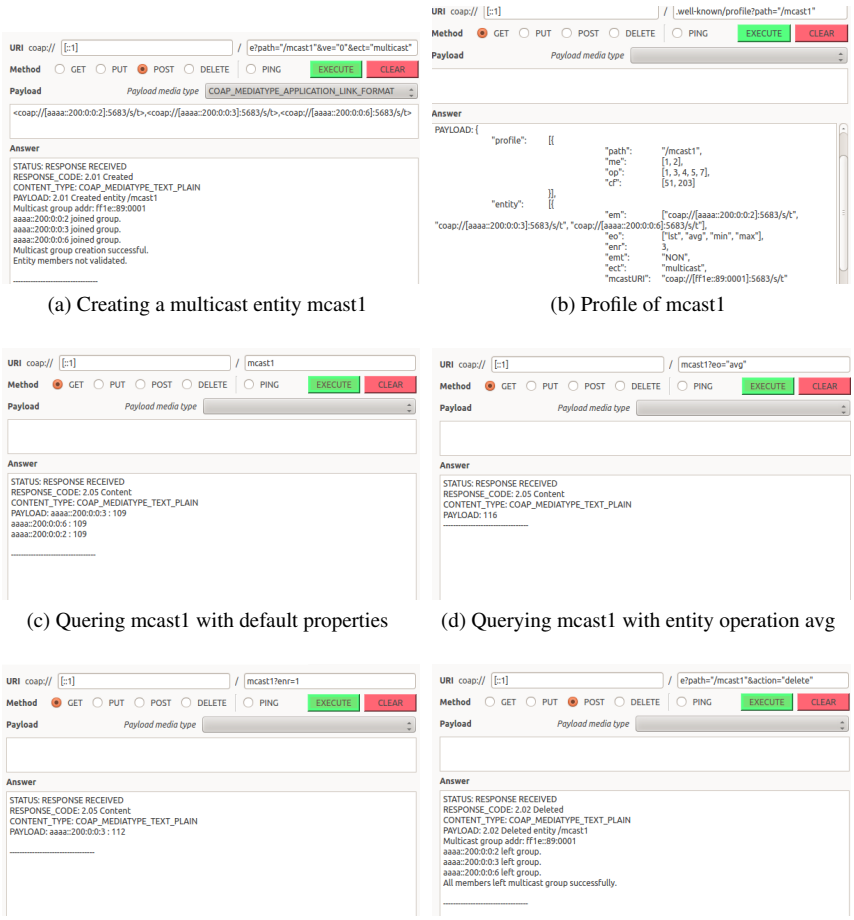
The functionality for creating, validating, using and deleting multicast entities has been implemented as described above. In this subsection we demonstrate the main functionalities of the implementation using a series of screenshots covering the life cycle of a multicast entity (Figure 5.3). These screenshots are taken using the CoAP++ client Graphical User Interface (GUI).

In Figure 5.3a the client initiates a request to create a multicast entity consisting of three members. The members are three temperature resources on RM090 sensors. The EM first assigns a unique IPv6 multicast address [ff1e::89:1] to the group and then asks all members to join this multicast group. The entity is created successfully and is ready for use since all members have reported that they have successfully joined the multicast group.

Next, the client checks the profile of the newly created entity (Figure 5.3b). The profile confirms that the entity uses multicast for communication (`ect:multicast`). Consequently, CoAP NON messages will be used to communicate with the members (`emt:NON`). The profile also shows that the entity supports four Entity Operations: `lst, avg, min` and `max` where lst is the default operation as it is the first operation in the list of supported operations. It further shows that the EM will wait until it receives 3 replies from the members before sending the combined reply to the client (`enr:3`).

Next, the client queries the newly created entity without specifying any Uri-Queries and thus the default entity properties (Entity Operation, Entity Number of Replies, etc.) are applied. Figure 5.3c shows how the client receives a reply containing all members values.

Now assume the client is not interested in the individual values of all members. As part of the request, the client uses the Entity Operation `avg` as part of the URI



(a) Creating a multicast entity mcast1

(b) Profile of mcast1

(c) Quering mcast1 with default properties

(d) Querying mcast1 with entity operation avg

(e) Using mcast1 as an anycast entity

(f) Deleting the entity mcast1

Figure 5.3: Screenshots using the CoAP++ client GUI to create, query and delete a multicast entity of three members.

query to obtain the average temperature of all three sensors. By doing so, the EM now calculates the average and only sends the calculated value to the client (Figure 5.3d). The disadvantages of this approach is that the EM will wait until it receives replies from all members before sending the reply to the client and that it might not get all replies since multicast communication is not reliable. Thus the client decides to set the Entity Number of Replies to a single reply (`enr=1`). By doing so, the EM will send a reply to the client as soon as it gets a reply from any of the members (Figure 5.3e). This way the EM emulates anycast behaviour which is not supported by RPL.

To complete the life cycle of this multicast entity we show how the client deletes it in Figure 5.3f. As a result of deleting the entity the EM requests that all members leave the multicast group, waits until the members confirm that they have left the group and conveys the results to the client. To avoid further confusion (e.g., bookmarked URIs containing the multicast addresses), the EM will not use the same multicast address for new groups unless it runs out of addresses.

5.5.1.2 Extended entity features

Our solution is flexible and can be used to build more complex entities than we have shown so far. Since entities have URIs and behave in the same way as any other CoAP resource, entities can themselves become part of other entities to build a hierarchy. To illustrate this consider the following scenario.

A large cold storage building consists of several floors with several cold storage rooms in each of them. In each room there are at least three battery operated CoAP enabled wireless temperature sensors. For quality assurance purposes, it is required to centrally log the average temperature of each room every hour.

The most straightforward solution is to query all these sensor individually and add all the intelligence about the distribution and the processing of the individual values in the client application. However, using our solution this can be accomplished in a hierarchical way as follows

1. Per room, create a **room temperature entity** containing all temperature sensors. Since every room has at least three sensors and is required to get the average temperature, the entity should use the Entity Operation `eo=avg`. Since the sensors are battery operated they are likely to have a Radio Duty Cycle (RDC) with long sleep periods. Further, as only the average is needed the administrator can decide for each room how many values are needed to build the average and set the Entity Number of Replies accordingly e.g., `enr=2`. To reduce the communication overhead, multicast communication with the sensors can be used i.e. `ect=multicast`.

2. Per floor, create a **floor temperature entity** containing all the room entities of the floor, i.e. as members the URIs of the previously created room temperature entities are used. Since it is expected to have values for all rooms, this entity can be created with default behavior which implies `eo=lst`, `enr="number of entity members"` and `ect=unicast`.

3. Create one **building temperature entity** containing all floor temperature entities in a similar way as the floor temperature entities.

4. Query the building temperature entity to get results for all the rooms in the building. The client can decide in which media type to get the reply. Using

plain text might be useful for the client to view or to include in reports. Since in this scenario the data should be logged, it is more likely to request the data in a more structured format such as SENML+JSON [25], so it becomes easier to extract and add the values to a database.

The example above can be easily extended in the hierarchy (e.g., building, campus, etc.). Entities created for one reason can be reused for other purposes, e.g., the room temperature entities can be used by the respective cooling system to control the temperature in rooms. This is true even if the entity should behave in a different way than it was created. For example, to troubleshoot heat distribution inside a room, a technician can query the same room temperature entity, but requesting to get a list of all individual sensor values in order to build a room heat map. From this example one can see that entities can be configured and combined in flexible ways to suit the clients needs.

5.5.2 Performance evaluation on a wireless sensor testbed

We have conducted the majority of the evaluation tests on the w-iLab.t wireless sensor testbed in Zwijnaarde[2]. This testbed provides a controlled test environment in an large ($66m \times 20m$) open room with 60 fixed nodes and 15 mobile nodes. Each node includes sensors (Rmoni RM090 and Zolertia Z1) and Wi-Fi (IEEE 802.11a/b/g/n). In our experiments we have used up to 40 fixed nodes as sensor nodes, two fixed nodes as sensor network GWs, and two other fixed nodes to generate interfering Wi-Fi traffic when needed. Figure 5.4 shows the part of the testbed that we used during our experiments. The other nodes were idle during the experiments to make sure that they do not cause extra interference and are not shown in Figure 5.4 for clarity. The two GWs are connected via an Ethernet link. When both GWs were used, they operated on different Zigbee Radio Frequency (RF) channels. However, in most experiments we used only GW1 and kept GW2 idle in order to not interfere with the running experiments. In all the performance evaluation experiments we used the Rmoni RM090 boards with Contiki. We have used RPL as routing protocol and enabled the SMRF multicast engine. We did not use any RDC. The GWs are running the example rpl-border-router provided by Contiki and therefor they are the RPL DODAG roots for their subnets, delegate the global IPv6 prefix and route traffic to and from the constrained networks. All other nodes run the Erbium server and also have RPL enabled. In Table 5.4 we summarize the most important settings for Contiki and the used protocols.

In the following subsections we present the results of our evaluation experiments on the testbed and compare them with simulated results when appropriate.

[2]http://ilabt.iminds.be/wilabt

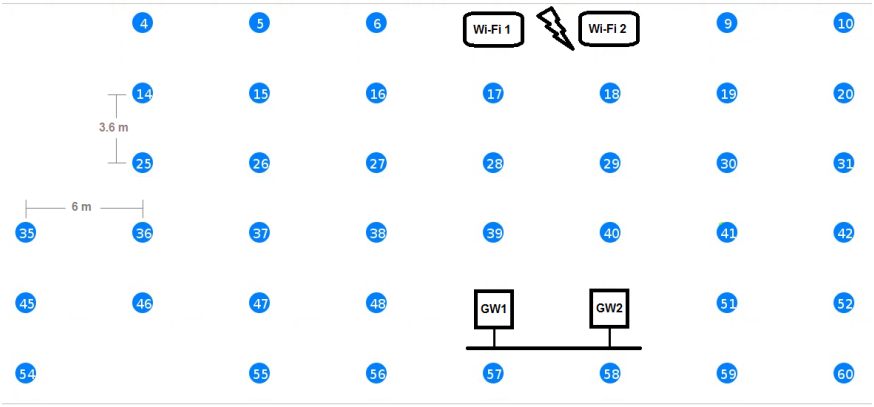


Figure 5.4: Experimental setup at w-iLab.t Zwijnaarde - Generic Wireless Testbed. The circles represent the location of the nodes.

5.5.2.1 Congestion control optimizations

An important aspect of group communication is congestion control, especially in LLN where resources are limited. Network congestion can lead to extended response times and significant energy consumption due to frequent retransmissions of packets. CoAP provides basic congestion control by using the exponential back-off mechanism (Section 5.3.1) and by limiting the number of open requests from a client to any server to one request by default. Furthermore, CoAP specifies that, when using multicasts, a certain random delay should be inserted before replying to multicast requests. In CoAP terms, this delay is called Leisure. The server could either use a default value for Leisure or compute a value for it. If the server has a group size estimate G, a target data transfer rate R and an estimated response size S, a rough lower bound for Leisure can then be computed as:

$$Leisure_{lowerbound} = \frac{S * G}{R} \tag{5.1}$$

In our experiments G was between 5 and 40, S equals approximately 80 bytes, and the target rate R can be set to a conservative 8 kbit/s = 1 kB/s. The resulting lower bound for the Leisure is then between 0.4s and 3.2s. However, since CoAP servers will often not be able to compute the Leisure, we elected to use the default Leisure value (5s) in all of our multicast experiments. For a more complete discussion of the Leisure period and its estimation we refer to Section 8.2 of [9].

CoAP does not specify a congestion control mechanism when a single client is communicating with many servers using unicasts as is the case in our group communication solution. However our experience shows that this can quickly lead to congestion. A simple solution for avoiding network congestion when using

Table 5.4: Evaluation experiments settings.

Node Type	Rmoni RM090	
Contiki version	Github master branch – snapshot 2 July 2014	
Radio Duty Cycling (RDC)	Null-RDC	
Media Access Control (MAC)	Carrier Sense multiple Access (CSMA)	
Routing	Protocol:	RPL
	Mode:	Storing
	Max neighbors:	50
	Max routes:	60
Multicast	Engine:	SMRF
	Max multicast addresses:	6
Radio Frequency (RF)	GW1:	Zigbee channel 26
	GW2:	Zigbee channel 25
	Wi-Fi 1 and 2:	Wi-Fi channel 13
CoAP	Version:	Draft-18
	Leisure:	5 seconds

unicasts is to limit the rate at which requests are sent. This way the group members will get the requests spread over a period of time and thus there replies will also be spread over a period of time in a similar way to Leisure. In order to get the replies spread over a period $Leisure$ the EM should insert a delay between requests D that equals $Leisure$ divided by $G-1$, e.g.:

$$D_{lowerbound} = \frac{Leisure_{lowerbound}}{G-1} = \frac{S*G}{R(G-1)} \tag{5.2}$$

For our experiments we get $D_{lowerbound} = 100ms$ and $D_{lowerbound} = 82ms$ for $G = 5$ and $G = 40$ respectively. In order to verify the effect of the delay length, we conducted a series of experiments on the testbed to query an entity of five members and measure the response time, which is expressed as the time between the moment the client issues the request to the EM until it gets back the response. We repeated the same experiment for different delays between the requests sent from the EM to the members. We repeated the experiment 50 times for each setting and computed the averages. During these experiments all Wi-Fi devices were turned off and thus no noticeable external interference was present. In [6], we have done the same experiments, but using the COOJA network simulator. Figure 5.5 shows the results of the experiments on the testbed and in COOJA. In general there is a very good match between the results of the simulation and the results on the testbed. The Figure clearly shows the need for the delay between the requests. Without inserting the delay the response time of the entity was about 3 seconds. When using a delay of 0.1s as calculated from equation 5.2, the response time drops to 550ms and is very close to the minimum value of 390ms that was

achieved for a delay of 50ms.

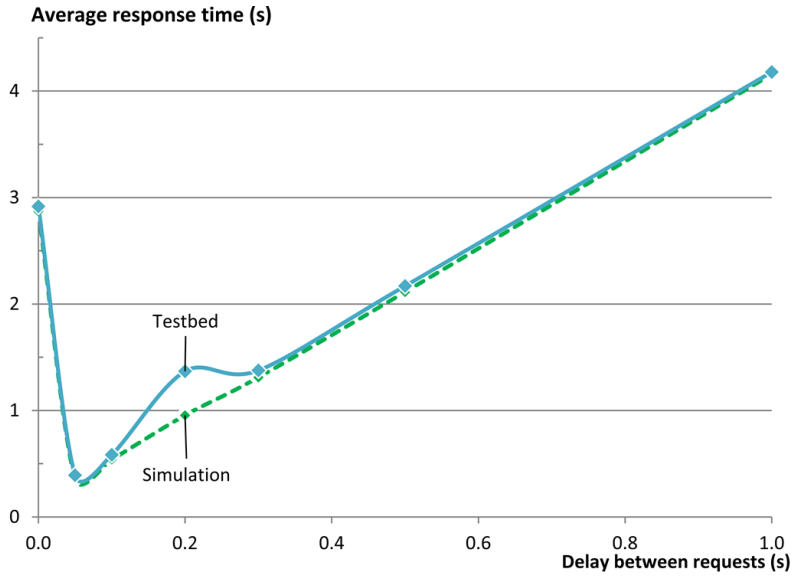


Figure 5.5: The Entity response time vs. the delay between the requests to the entity members as measured on the testbed and in COOJA network simulator for an entity of 5 members.

In order to verify whether the same relationship exists between the delay and the response time for other group sizes, we have repeated the same set of experiments on the testbed using additional group sizes ($g = 10, 20, 29, 40$). Figure 5.6 shows the results of those experiments. Since the EM sends the requests to the members sequentially, it is expected that the response time for the complete entity gets larger as the group size gets larger. This relationship is very obvious in the Figure. Regardless of this fact, one can see that graphs for all the group size follow the same pattern.

To further analyze the relationship between the delay and the group size, please consider Figure 5.7, which shows the same results as Figure 5.6, but this time normalized over the group size G. In a star topology such as in our case, where all members need to communicate with the root of the star (the GW), one expects that the average response time would increase as the number of neighbors increases, resulting in a higher number of collisions on the shared medium. This is indeed the case when we compare any larger group size with the group of five members. However when comparing the larger groups together, this relationship cannot be observed. The reason for this is that with the increase of the group size and the way nodes are distributed over the testbed, members become no longer directly reachable from the GW and their traffic was routed via other members. Consequently,

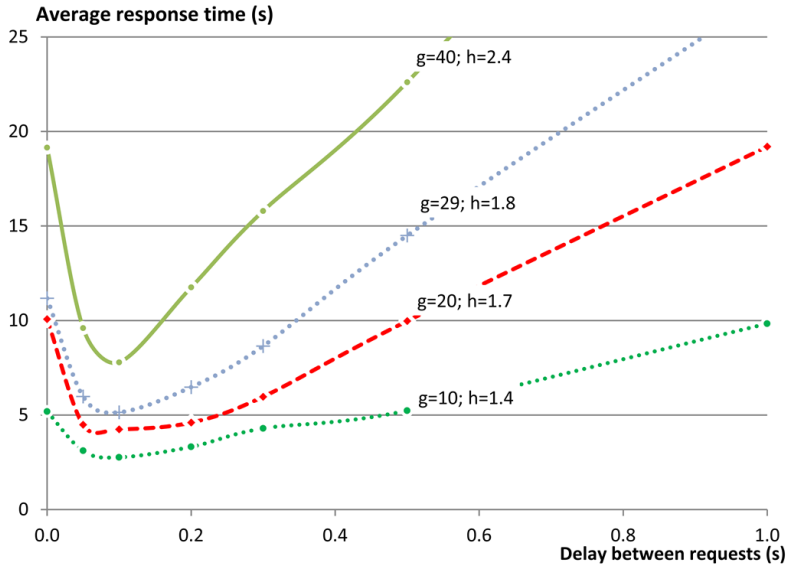


Figure 5.6: The Entity response time vs. the delay between the requests to the entity members as measured on the testbed for different group sizes.

Table 5.5: Characteristics of the WSNs used in congestion control experiments on w-iLab.t Zwijnaarde wireless sensor testbed.

Number of Nodes	Hop Count	
	Average	Maximum
5	1	1
10	1.4	2
20	1.7	3
29	1.8	3
40	2.4	5

the average hop count was also increasing from just one hop for the group of five members to 2.4 hops for the group of 40 members while the maximum hop count increased from 1 to 5 (see Table 5.5). A higher hop count implies that a lower percentage of members can communicate directly with the GW. It also means that a lower percentage of nodes is in the collision domain of the GW. This makes it possible that more parallel communication can happen inside the Wireless Sensor Network (WSN) before they reach the collision domain of the GW where the bottleneck is located.

Regardless of the small changes in the values for the various groups, we observe that the shape of the relationship function is very similar among all of them. The response times were always improved significantly when the delay was around

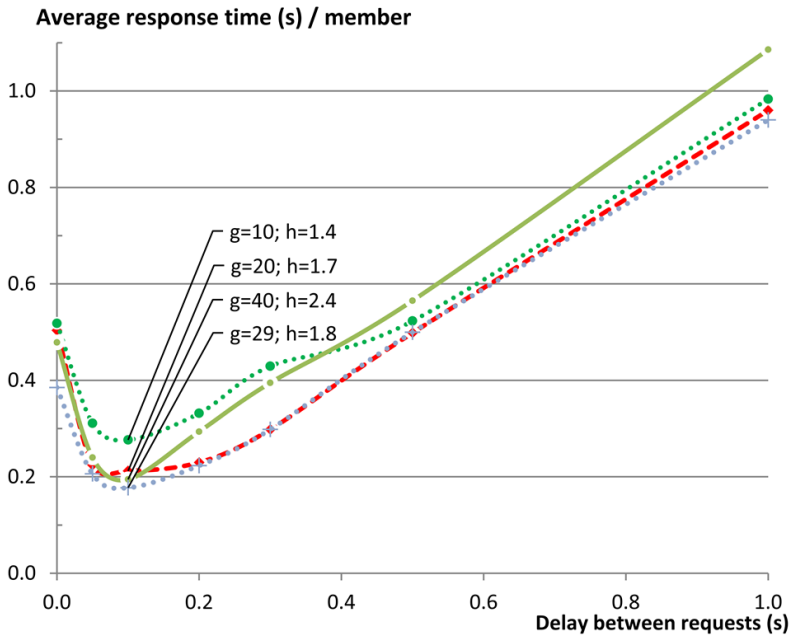


Figure 5.7: The Entity response time per member vs. the delay between the requests to the entity members as measured on the testbed for different group sizes.

the recommended range of 82ms to 100ms. However as the delay between requests grows larger it becomes the dominating factor for the total response time with a linear relationship between the two.

Another indicator of the performance of any communication solution is its reliability. During the tests we conducted in this section the reliability of the communication was always 100% for all group sizes lower than 40. This is not surprising since we had not external interference and the only cause for errors was internal collisions. For the group size of 40, the reliability of member replies was never 100%. It was always between 99.8% and 99.9%, regardless of the delay between requests. This is also not surprising as with the larger group size, the chance for collisions increases and the CoAP retransmission mechanism starts to be sometimes insufficient. We will discuss reliability in more detail in the next subsection.

As a result of the observations we made in these experiments we have used an entity delay between requests of 100ms in all following experiments, which is also in line with the results of equation 5.2.

5.5.2.2 Reliability

Reliability is a key performance indicator. In this subsection we experimentally evaluate the reliability of both unicast and multicast CoAP group communication in the presence of Wi-Fi interference. To generate this interference we send UDP traffic from one Wi-Fi node to the other at a constant bandwidth by using the `iperf` tool. We have setup the Wi-Fi communication to use Wi-Fi channel 13, which completely overlaps with Zigbee channels 25 and 26 that we use inside the WSN. Since we are using CSMA as Media Access Control (MAC), the sensor nodes will back off when Wi-Fi is sending. However this is not true for the other direction. Typically, Wi-Fi MAC will not detect that wireless sensors are sending and will not back off.

To measure the reliability we used the same experiment setup shown in Figure 5.4 to communicate with a group of 10, 20 and 30 members. We gradually increased the Wi-Fi interference in the network in steps of 5Mb/s and measured the reliability of getting responses to the respective requests. We repeated the same experiment for our group communication solution and for multicasts. We run each experiment 50 times and show the averages in Figure 5.8. Multicasts are not transported reliably and thus reliability of the network decreases as soon as there is packet loss due to the Wi-Fi interference in the network. When using our unicast group communication solution, CoAP confirmable messages are used.

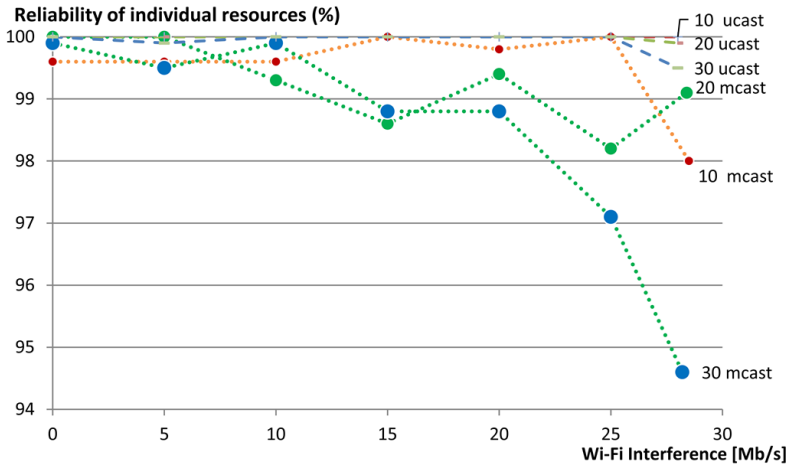


Figure 5.8: The reliability of individual group members is a lot better when using unicast based group communication.

For the group of 10 members, reliability of individual resources remains always 100% even when the Wi-Fi nodes were transmitting as fast as they could (28.5 Mb/s). The reliability of individual resources for the group of 20 nodes

dropped a bit to 99.9% under maximum Wi-Fi interference. For the 30 members group the reliability is further reduced to 99.5% (compared to 94.6% in the case of multicasts). Figure 5.8 also shows that the reliability of individual members decreases with an increasing group size, both for unicast and multicast communication. This is due to two reasons. First, larger groups are denser and thus have a higher chance of collision between the group members. Second, and maybe with a higher impact on the reliability, bigger groups have a larger average hop count. This means that every message (both request and reply) between a client and a server has an additional chance of getting dropped at each hop on the way to its destination. Nevertheless, in our 20 and 30 members groups 100% reliability was maintained for unicast communication until a Wi-Fi transmission rate of 25Mb/s with one single exception for the 30 members group at 5Mb/s, where 1 message was lost and resulted in a reliability of 99.9%.

In many group communication use cases, it is desirable to get answers from all members of the group. A complete group communication is considered successful when communication to all members in the group is successful. Figure 5.9 shows the effect of packet loss on the reliability of the complete group for our 10, 20, and 30 members groups. Certainly the reliability of a complete group is less than the reliability of its individual members, since the loss of a message to or from a single member, renders the complete group request unsuccessful. In these cases the use of multicasts does not provide good results. Already at 15Mb/s Wi-Fi traffic the reliability of 20 and 30 members groups drops to about 80%. In contrast, our unicast based group communication maintains 100% reliability for the 10 and 20 members groups, even with the maximum transmission speed of the Wi-Fi nodes, and only drops to 98% in the case of 30 members group.

These results are generally in line with the simulations that we performed previously in [6]. However direct comparison is not possible, since the simulations used a more controlled topology, in which 5 nodes were 1-hop away, another 5 nodes were 2 hops away and so on. On the testbed, the location of the nodes is fixed and it was up to RPL to construct the topology for the routing. Also the simulations randomly dropped packets at a configurable percentage to simulate external interference, while on the testbed real Wi-Fi traffic at one point of the network was used.

5.5.2.3 Response time

Another key indicator of the performance of any group communication solution is its response time. Figure 5.10 shows that the average response time when using our group communication solution increases with the increase of the Wi-Fi interference in the WSN. When there is no loss, the response time for unicast is just above the sum of the delays between the requests that the EM added, i.e. 1, 2 and 3 seconds for the groups of 10, 20 and 30 members respectively. For multicast the

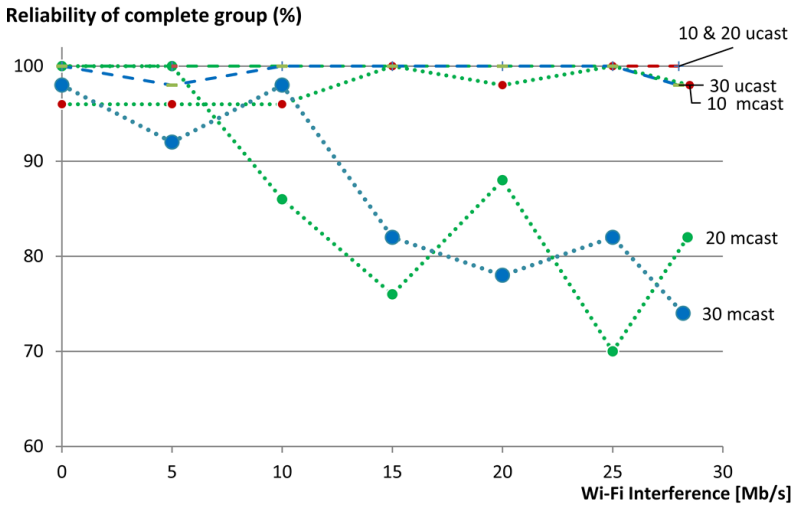


Figure 5.9: The reliability of the complete group is less than the reliability of individual members (Figure 5.8). Again, the reliability of the complete group is a lot better when using entity based group communication.

response time is determined by the Leisure (5s in our case). It is always lower than 5s, since the nodes choose a value between 0 and 5s before sending their replies. The value that is shown here is the value until the last response was received at the EM. When the interference increases, the response times for multicasts remain almost unchanged, since either the packet is delivered on time or it is just dropped. This way, the multicast solution is capable of maintaining lower response times at the expense of a decreased reliability. In the case of our solution, when a packet is dropped CoAP attempts to retransmit it, leading to an increased overall response time.

These experimental results are also in line with the simulated results in [6]. Again a direct comparison with the simulated results is not possible as explained at the end of Section 5.5.2.2.

5.5.2.4 Group size

As shown in the previous subsections using large groups can have a negative impact on the reliability of the group. In our tests unicast groups started to become unreliable after a group size of 30 members. Multicast groups are generally unreliable but the reliability also becomes worse with an increasing group size. The reason for this is that with the increase in group size, the density of the nodes typically also increases and as a result more collisions occur in the network. Also for the unicast-based solution, the group size directly affects the response time since

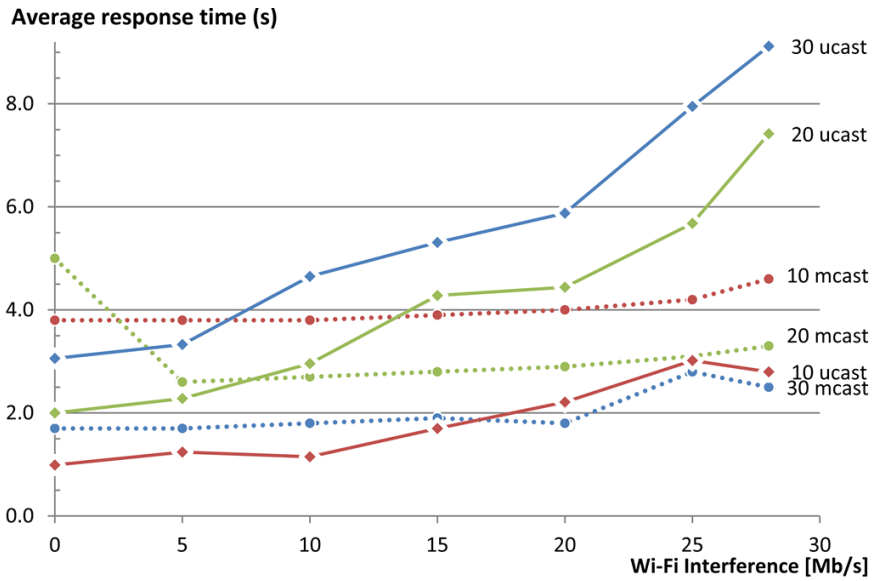


Figure 5.10: Average group response time.

the EM adds a delay between the requests it sends to the members. One simple solution is to split the groups. However, splitting the groups does not bring a lot of benefit, when both groups are still using the same RF channel. When using our group communication solution one can use more than one GW and create different WSNs that use different Zigbee RF channels. The groups are split accordingly.

In order to test this approach and to demonstrate the use of more than one GW to create two WSNs that are overlapping in the physical space but are using different RF channels, we have created a new experiment. In this experiment each GW is communicating with a network of 10 sensor nodes using its own RF channel (Zigbee channels 25 and 26). The two GWs are connected via an Ethernet cable and routing is enabled between them. We have repeated the same test as in Section 5.5.2.3 but now using a group that consists of two smaller groups. Figure 5.11 shows the response time vs. the speed of interfering Wi-Fi traffic for the new experiment along with the results for groups of 10 and 20 members from the previous section for comparison. As expected, the response time for the group of two smaller groups is better than that of the one big group, although the total number of nodes was the same in both cases (20 nodes). Further, the response time is larger than the case of a single group with 10 members. The reason here is that we use nested groups, i.e. a group that contains two groups. This results in some additional processing overhead and also inserts a delay of 100ms between the requests being issued to the different subgroups. Additionally, since we used two

neighboring Zigbee channels, a small amount of interference between the channels is present. The reason for selecting two neighboring channels was to have both channels equally interfered from the Wi-Fi channel 13, that overlaps both of the used Zigbee channels 25 and 26. In a production setting, one should not use a neighboring channel to also avoid this limited amount of interference. The selection of channels should also take into consideration which Wi-Fi channels are used.

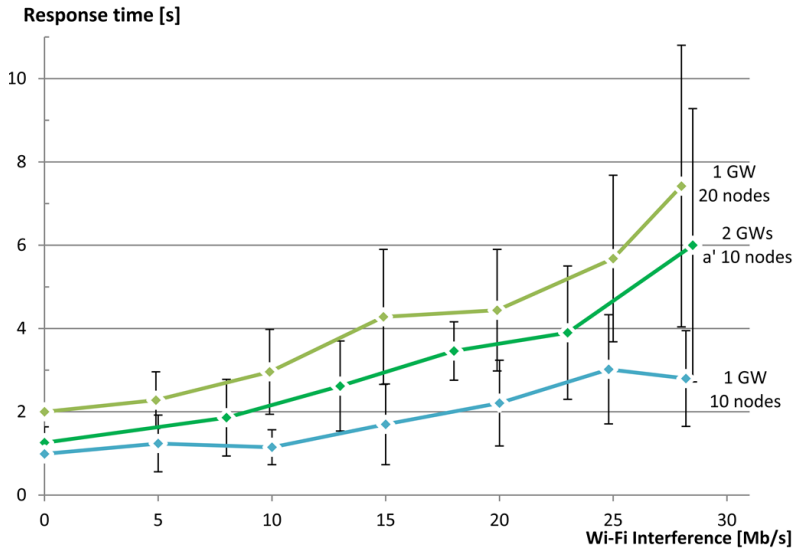


Figure 5.11: The Entity response time vs. the speed of interfering Wi-Fi traffic for an entity of 20 members vs 2 entities of 10 members each.

5.5.2.5 CoAP retransmission timeout

As described in Section 5.3.1, CoAP has its own basic reliability mechanism that can be used for unicast communication. When reliability is needed, the sender of the CoAP message should use a Confirmable Message (CON). The receiver has to acknowledge this type of messages by sending an ACK. If the sender does not receive a reply within a back-off time, it retransmits the Confirmable message at exponentially increasing intervals, until it receives an ACK or runs out of attempts. By default the initial back-off is set to a random time between `ACK_TIMEOUT` and `ACK_TIMEOUT * ACK_RANDOM_FACTOR`. By default, `ACK_TIMEOUT`=2s, and `ACK_RANDOM_FACTOR`=1.5 and thus the default initial back-off is between 2 and 3s. If a reply to the first transmission attempt of a CON is not received within the initial back-off time, the CoAP sender will double the initial back-off time and retransmit the packet. If a reply to the first retransmission is not received,

CoAP will again double the back-off time and retry the transmission until MAX_RETRANSMIT (by default 4) is reached. If no reply is received after expiration of the back-off time of the last retransmission, the client will be notified about the error condition. When using the default values, the best case timeout will be after $2+4+8+16+32 = 62$ seconds and in the worst case after $3+6+12+24+48 = 93$ seconds.

The CoAP protocol allows the client to change the default parameters according to its needs. Changing those parameters will effect both the reliability and the response time. Changing MAX_RETRANSMIT effects the reliability directly, since it changes the number of attempts to get a successful communication. In our tests the reliability was most of the times 100% with the exception of using large groups and large interference. As such, the default value of 4 retransmissions is fine for our use case. On the other hand, changing ACK_TIMEOUT, and thus the initial back-off time, has a direct impact on the response time, since it specifies the time between the retransmission attempts. In order to investigate the effect of changing the initial back-off time on our solution we have conducted a series of tests that are similar to those described in Section 5.5.2.2 for different values of the initial back-off times. Figure 5.12 shows the effect of Wi-Fi interference on the response time for three different values of the initial back-off time (ACK_TIMEOUT = 0.5 , 1 , and 2s) for a group of 10 members. When there is no Wi-Fi interference there is no need for retransmissions and thus the initial back-off has no effect. When Wi-Fi traffic was interfering with our WSN, reducing ACK TIMEOUT from 2 seconds to 1 second helped to improve the response time. However, reducing ACK TIMEOUT further to 0.5s had a negative effect. This is due to the fact that in this case CoAP was not waiting long enough for the replies to arrive and meanwhile trying to retransmit the requests, causing more collisions in the network.

5.5.3 Evaluation in a real life setup

The evaluations we have conducted so farwere performed in a more or less controlled environment in an open-room testbed. This allowed us to analyze the behavior of the different group communication approaches in a clean environment with very limited external interferences. It also allowed us to inject controlled Wi-Fi interfering traffic and study its effect of the WSN and the used group communication approaches. By doing so we were able to compare the obtained results with the simulation results, which were also conducted in a comparable controlled environment. In some use cases, the radio environment might be clean, such as is the case inside a shielded room such as our testbed. However, in many use cases, and in our building automation use case in particular, it is expected that a lot of interference will be present that will affect the WSN. Typical WSNs operate in the Industrial, Scientific and Medical (ISM) radio bands. These radio bands are used

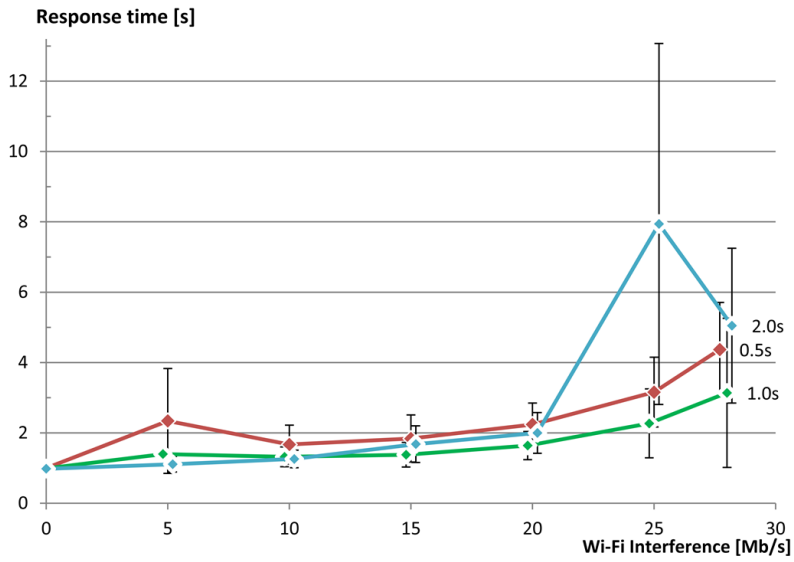


Figure 5.12: Effect of the Wi-Fi interference on the response time for three different values of the initial back-off time (`ACK_TIMEOUT` = 0.5 , 1 , and 2s) for a group of 10 members.

by many other devices, not just WSNs and Wi-Fi, e.g., microwave ovens, cordless phones and Bluetooth.

In order to evaluate the behavior of our solution and that of the standard multicast solution in an environment that is more realistic than the open-room testbed, we have conducted a series of tests on the w-iLab.t office testbed. This wireless sensor testbed contains about 200 nodes spread over 3 floors (15 by 90 meters each) of an operational office building in Ghent, Belgium [26]. In our experiments we used 10 nodes spread over parts of the third floor during office hours (Figure 5.13). The circles represent the location of the nodes at the 3rd floor of the office building. The filled circles represent the nodes used in the experiment. The other nodes were idle. As can be seen in the Figure, the nodes are located in different rooms and in the hallways. In this setup we have no control over the amount of interference that occurs during the experiments since the sources of interference are diverse and are beyond our control – as it is typically the case when using the Industrial, Scientific and Medical (ISM) radio band. A main source of interference in this part of the building is Wi-Fi, which was operating on Wi-Fi channel 13. As explained before, this channel completely overlaps Zigbee channel 26 that we used inside the WSN. Other sources of interference are also present (cordless phones, bluetooth headphones and a microwave oven) but those are not always used.

We conducted two sets of experiments at two different times of the day in order to get different conditions for the tests. The first set of experiments was conducted

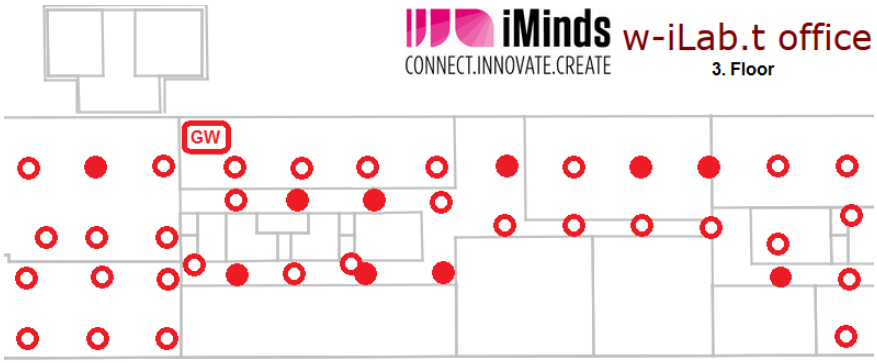


Figure 5.13: Experimental setup at w-iLab.t office. The circles represent the location of the nodes in the 3. floor of the office building. The filled circles represent the nodes used in the experiment. The other nodes were idle.

Table 5.6: Summary of the results of the experiments on w-iLab.t office wireless sensor testbed.

Communication Type	Network Usage	Reliabilty		Response Time [s]		
		Member	Group	Min.	Avg.	Max.
Unicast	Low	100%	100%	1.02	1.70	4.53
Unicast	Normal	100%	100%	2.91	7.21	13.87
Multicast	Low	92.2%	64%	2.62	4.92	8.36
Multicast	Normal	91.2%	58%	2.45	4.79	7.75

when a few people were present at the office and the other set was conducted when almost everybody was present and working. In each set of experiments we queried the group of ten nodes 50 times using our unicast-based group communication solution and another 50 times using multicast. Similar to the experiments in the open-room testbed, we have measured the reliability of the communication and the response time. In Table 5.6 we summarize the results of these experiments. These results confirm the results we obtained from the experiments on the open-room testbed. Under *normal* network environments, multicasts reliability is not very encouraging. Replies were received from the members with a success rate of 91.2%, resulting in a poor 58% reliability for the complete group. On the other hand the reliability of our solution was maintained at 100% since the CoAP retransmission mechanism was able to handle all errors. Of course this leads to higher response times for the unicast groups, which increased from 1.7s to 7.21s on average between the two experiments. As expected the response time for multicasts was not much affected.

In order to take a deeper look at the group response times we summarize them

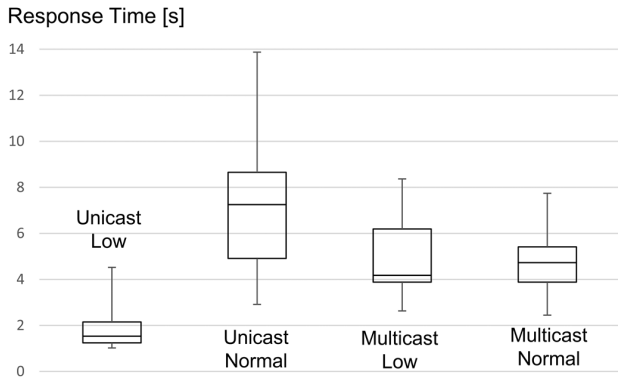


Figure 5.14: Tukey boxes with whiskers summarizing the group response times for unicast and multicasts groups of ten members on w-iLab.t office testbed. Unicast reliability is achieved by exponential retransmissions and lead to large increase in the response times.

in Figure 5.14 as Tukey boxes with whiskers and a central point [3]. Because of the well-defined back-off mechanism used by CoAP, it is possible to extract information about the amount of retransmissions, and thus the reliability, from a Tukey boxplot of the latency. Looking at the box for unicast under low network usage, one can say that roughly in 70% of the cases all group members replied after the first request, since the EM replied in less than 2s and the first retransmission occurs after 2s to 3s. For the remaining 30%, at least one member of the group needed one retransmission. A second retransmission, taking place between 6s and 9s, was never needed. In the case of multicast communication, the latency is significantly higher because of the way we configured the leisure period. Here, every packet loss will directly result in a decrease of the group reliability. Under low network usage, the reliability dropped to 62%, which is more or less in line with the packet losses observed during the unicast experiments, which was executed under similar conditions. Now, looking at the box for unicast under normal network usage we see that less than 25% of the cases required no retransmission. For the remaining 75% of the cases, one or two retransmissions were needed for at least one member of the group. A third retransmission, which would occur between 14s and 21s, was never needed.

5.6 Related work

As mentioned in the introduction, the basis of our work is the newly standardized Constrained Application Protocol (CoAP) [9] and the two main approaches to

[3] Tukey boxes, with whiskers and a central point, are five-point summaries of data sets, corresponding to the minimum, 25th percentile, median, 75th percentile, and maximum.

realize CoAP group communication.

The first approach been published as an experimental standard for CoAP group communication [5] by the IETF CoRE working group, who was behind the standardization of CoAP itself. This standard specifies how CoAP should be used in a group communication context. It provides an approach for using CoAP on top of non-reliable IP multicast. Certainly, the use of multicasts allows reducing the amount of requests in the LLN, by sending one request to several destinations at the same time. However, the use of multicast has limitations such as not being cache-friendly and not supporting secure communication (see Section 5.3.2).

An alternative multicast-based solution was presented in [27]. This work presented a concept of a Web service based communication stack that uses transient link-local IPv6 multicast addresses for process data exchange between nodes by binding a shared network variable to the IPv6 multicast address. By doing so, the authors were able to reduce the CoAP multicast size by eliminating the need to include the URI path to identify the group communication resource on the members. URI identifiers are text-based. As such, they can be verbose and have a severe impact on the CoAP header size, since no compression can be provided for them. Certainly this approach has several advantages such as eliminating the need for a control unit, offering a lower power consumption than using unicasts and its suitability in many non-critical use cases (due to the lack of reliability of multicasts). However, since this approach is based on IP multicast, it exhibits the limitations of multicasts as discussed in Section 5.3.2. In addition, to our knowledge, apart from simple performance evaluations, no larger scale evaluations of the performance of multicast solutions have been published. Consequently, at of the time of writing, clear insights in the behavior and performance of multicast operations in LLNs were missing.

The second approach is to rely on unicast messages to realize CoAP group communication. In [6] we have introduced a unicast-based group communication solution that uses the notation of entities to represent groups of CoAP resources. We have evaluated it using simulations and small-scale demonstrations and showed that it complements multicast-based solutions when reliability of the communication is desired. Another unicast-based group communication solution is called SeaHttp and was presented in [28]. The authors propose to extend CoAP with two additional methods (BRANCH and COMBINE) to allow members to join and leave groups without the need for a separate group manager. This means that members should have the intelligence to know which group they should join/leave. Constrained devices will not have this intelligence, so again, a manager will be needed to inform the devices so they can take appropriate actions. Furthermore, BRANCH and COMBINE can maybe reduce the number of messages; however, the trade-off is the need to implement a new mechanism. It is better to use an approach that can be plugged in into any existing network without major modifi-

cations (or at least not a modification to every node). Finally, this approach does not have the flexibility we target, since group members have to be reprogrammed with the groups they should join each time the requirements of the user changes.

To our knowledge, these are the only works that explore communication solutions for interacting with a group of CoAP-enabled constrained devices. Next to these, there exist other solutions to realize group-like communication in constrained environments without using CoAP. For example, the Message Queue Telemetry Transport (MQTT) protocol is another application layer protocol designed for constrained devices [29]. MQTT uses a topic-based publish-subscribe architecture, i.e. clients utilize the services of a *Broker* to subscribe to *Topics* and get all the *Messages* that are published to that topic. Unlike CoAP, MQTT relies on TCP as underlying transport protocol and thus inherits its reliability. While the base CoAP does not provide any Quality of Service (QoS), MQTT provides its own QoS mechanism. MQTT provides its own way of group communication, by allowing multiple publishers to publish to the same topic and by allowing multiple clients to subscribe to it. This can be seen as a form of group communication which exhibits some similarities with our proposed approach. However, it does not adhere to the REST principles that are commonly used in the Internet and, additionally, it does not provide the possibility to aggregate and manipulate notifications that are sent to the clients.

5.7 Conclusions and outlook

The ability to communicate with groups of resources is important for many IoT applications in general and for BASs in particular. CoAP, which is expected to play an important role as an application protocol for use in constrained environment, does not have built-in group communication features. However, CoAP is designed in way that makes it easy to extend. Currently there are two main approaches to extend CoAP with group communication capabilities. The fundamental difference between the two approaches lies in the underlying communication type: multicast versus unicast. The trade-off between the two communication types is reliability versus speed. In this paper we proposed a hybrid solution that tries to get the benefits of both approaches. The solution is flexible to allow the user to select the communication type based on the desired features. As such, we believe that our solution is a powerful enabler for group communication in LLNs and an interesting building block for IoT applications.

Next to this, we have experimentally evaluated those approaches using two WSN testbeds (one testbed in a shielded room and another in an office environment). We explored several aspects (overhead, timing, scalability, etc.) related to the usage in realistic sensor networks and compared the different approaches. Experimental evaluation reveals that there are limitations to the size of the groups.

This may impact the design of real BASs that typically operate in conditions where interference is inevitable. A further outcome of the experimental evaluation is the impact of the CoAP parameter settings (Leisure, back-off, etc.) on the group communication performance.

During the evaluations we have identified a number of possible paths for further optimizations and improvements. First of all, more in depth research on different back-off strategies and congestion control mechanisms is needed in order to achieve an optimal balance between reliability and latency. Further, we believe that tighter integration with lower layers may further improve latency and reliability. Depending on the configuration of the groups and the type of resources involved, application requirements can be derived and translated into optimal configuration of the lower layers – e.g., optimized MAC schedule, Time Division Multiple Access (TDMA), IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH), local retransmissions, etc. This way, powerful IoT application enablers at the higher layers can be combined with advancements at the lower layers, together delivering the performance expected by IoT applications and their users.

Acknowledgement

The research leading to these results has received funding from a VLIR PhD scholarship to Isam Ishaq.

References

[1] N. Kushalnagar, G. Montenegro, and C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. RFC 4919 (Informational), August 2007. Available from: http://www.ietf.org/rfc/rfc4919.txt.

[2] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550 (Proposed Standard), March 2012. Available from: http://www.ietf.org/rfc/rfc6550.txt.

[3] *Constrained RESTful Environments (core)*, 2014. [Online; accessed 19 March 2014]. Available from: http://datatracker.ietf.org/wg/core/.

[4] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. V. d. Abeele, E. D. Poorter, I. Moerman, and P. Demeester. *IETF Standardization in the Field of the Internet of Things (IoT): A Survey*. Journal of Sensor and Actuator Networks, 2(2):235–287, 2013.

[5] A. Rahman and E. Dijk. *Group Communication for the Constrained Application Protocol (CoAP)*. RFC 7390, IETF, October 2014. Available from: http://www.ietf.org/rfc/rfc7390.txt.

[6] I. Ishaq, J. Hoebeke, F. Van den Abeele, J. Rossey, I. Moerman, and P. Demeester. *Flexible Unicast-Based Group Communication for CoAP-Enabled Devices*. Sensors, 14(6):9833–9877, 2014.

[7] *Constrained RESTful Environments (core) – Charter for Working Group*, 2014. [Online; accessed 9 March 2014]. Available from: http://datatracker.ietf.org/wg/core/charter.

[8] C. Reinisch, W. Kastner, and G. Neugschwandtner. *Multicast communication in wireless home and building automation: ZigBee and DCMP*. In Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, pages 1380–1383. IEEE, 2007.

[9] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252, IETF, June 2014. Available from: http://www.rfc-editor.org/rfc/rfc7252.txt.

[10] W. Colitti, K. Steenhaut, and N. De Caro. *Integrating wireless sensor networks with the web*. Extending the Internet to Low power and Lossy Networks (IP+ SN 2011), 2011.

[11] A. Dunkels et al. *Efficient application integration in IP-based sensor networks*. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pages 43–48. ACM, 2009.

[12] K. Hartke. *Practical Issues with Datagram Transport Layer Security in Constrained Environments*. Internet-Draft draft-hartke-dice-practical-issues-01, IETF, April 2014. Available from: http://www.ietf.org/internet-drafts/draft-hartke-dice-practical-issues-01.txt.

[13] J. Hui and R. M. Kelsey. *Tulticast Protocol for Low Power and Lossy Networks (MPL)*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-ietf-roll-trickle-mcast-04.

[14] G. Oikonomou, I. Phillips, and T. Tryfonas. *Ipv6 multicast forwarding in rpl-based wireless sensor networks*. Wireless personal communications, 73(3):1089–1116, 2013.

[15] B. Greevenbosch, J. Hoebeke, I. Ishaq, and F. V. den Abeele. *CoAP Profile Description Format*, 2014. [Online; accessed 3 June 2014]. Available from: http://tools.ietf.org/html/draft-greevenbosch-core-profile-description-02.

[16] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester. *Facilitating sensor deployment, discovery and resource access using embedded web services*. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, pages 717–724. IEEE, 2012.

[17] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. *The Click modular router*. ACM Transactions on Computer Systems (TOCS), 18(3):263–297, 2000.

[18] *Zolertia Z1 Platform*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.zolertia.com/ti.

[19] *Rm090 — RMONI wireless nv*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.rmoni.com/en/products/hardware/rm090.

[20] *Contiki: The Open Source Operating System for the Internet of Things*, 2015. [Online; accessed 27 February 2015]. Available from: http://www.contiki-os.org/.

[21] *Erbium (Er) REST Engine and CoAP Implementation for Contiki*, 2015. [Online; accessed 27 February 2015]. Available from: http://people.inf.ethz.ch/mkovatsc/erbium.php.

[22] R. Hinden and S. Deering. *IP Version 6 Addressing Architecture*. RFC 4291 (Draft Standard), February 2006. Updated by RFCs 5952, 6052, 7136, 7346, 7371. Available from: http://www.ietf.org/rfc/rfc4291.txt.

[23] T. Narten, E. Nordmark, and W. Simpson. *Neighbor Discovery for IP Version 6 (IPv6)*. RFC 2461 (Draft Standard), December 1998. Obsoleted by RFC 4861, updated by RFC 4311. Available from: http://www.ietf.org/rfc/rfc2461.txt.

[24] F. Van den Abeele, J. Hoebeke, I. Ishaq, G. K. Teklemariam, J. Rossey, I. Moerman, and P. Demeester. *Building embedded applications via REST services for the Internet of Things*. In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, page 82. ACM, 2013.

[25] C. Jennings, Z. Shelby, and J. Arkko. *Media Types for Sensor Markup Language (SENML)*. Internet-Draft draft-jennings-senml-10, IETF, October 2012. http://tools.ietf.org/html/draft-jennings-senml-10. Available from: http://tools.ietf.org/html/draft-jennings-senml-10.

[26] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester. *The w-iLab. t testbed*. In Testbeds and Research Infrastructures. Development of Networks and Communities, pages 145–154. Springer, 2011.

[27] M. Jung and W. Kastner. *Efficient group communication based on Web services for reliable control in wireless automation*. In Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE, pages 5716–5722. IEEE, 2013.

[28] C.-D. Hou, D. Li, J.-F. Qiu, H.-L. Shi, and L. Cui. *SeaHttp: A Resource-Oriented Protocol to Extend REST Style for Web of Things*. Journal of Computer Science and Technology, 29(2):205–215, 2014.

[29] *Message Queue Telemetry Transport MQTT*, 2015. [Online; accessed 2 March 2015]. Available from: http://mqtt.org/.

6
Conclusion

"It always seems impossible until its done."

– Nelson Mandela (1918 - 2013)

Almost every day we hear about a new application of the Internet. More and more these applications involve everyday objects that we do not typically consider as connected devices, such as the refrigerator, the coffee machine and the wristwatch. As a result, the number of devices connected to the Internet is expanding rapidly and is expected to reach a number that is a magnitude higher than the population of earth in just a few years from the time of writing these words. The resulting Internet is called the Internet of Things (IoT) to stress the fact that it is connecting all sorts of *things*. Many experts expect that the majority of these things will soon be embedded devices with very constrained resources, i.e. limited memory, processing capabilities and power. These constraints in the resources are mostly driven by the need to reduce the cost and the energy consumption of these devices.

The integration of these constrained devices into the IoT brings new challenges for the research and the industry communities, mainly because these devices are so constrained that they cannot run standard Internet protocols, not to mention running applications on top of them in a secure manner. In order to overcome this hurdle, several companies developed specialized protocols that allowed connecting the respective companies' constrained devices into the Internet, one way or the other. Building upon those proprietary protocols, many vertical solutions were developed. Communication between these solutions is often very difficult or not

possible at all, which is not in the spirit of the Internet. In order to resolve this situation, Internet standardization bodies started developing open standard protocols to cover the various layers of the networking stack. One of the recent open standards is the Constrained Application Protocol (CoAP). CoAP is a constrained environments alternative to the Hypertext Transfer Protocol (HTTP) – the leading application layer protocol.

The goal of this PhD research was to develop a set of enablers on top of CoAP that make the integration and usage of constrained devices in the IoT easier. The remainder of this final chapter summarizes the most important work and the main conclusions of the performed research. We give a general conclusion on CoAP group communication and close with some directions for future work.

6.1 Summary and conclusions

In this dissertation we have described a novel self-organization solution to facilitate the deployment of sensor networks and enable the discovery, end-to-end connectivity and service usage of newly deployed sensor nodes. The proposed approach makes use of embedded web service technology, i.e. the Internet Engineering Task Force (IETF) CoAP protocol. By combining it with Domain Name System (DNS) and foreseeing HTTP-to-CoAP proxy functionality, it complies with current Internet standards. Automatic hierarchical discovery of CoAP servers is one of the key features, resulting in a browsable hierarchy of CoAP servers, up to the level of the sensor resources. By creating a hierarchy of linked CoAP servers, scalability can be addressed. At this point, existing web crawler solutions can be used to index and publish the sensor information to the outside world. As such, the proposed approach provides a feasible and flexible solution to achieve hierarchical self-organization with a minimum of pre-configuration. The solution is based on a minimal number of assumptions regarding the pre-configuration. With some additional improvements and the development of management tools, it provides a valuable contribution to facilitate the deployment of and access to sensor networks. As such, it represents an important building block facilitating the actual usage of embedded web services as is required for building the Web of Things. Once end-to-end access has been realized, adding new functionalities or building novel services involving IoT objects is straightforward and many opportunities to integrate this with existing web service technologies arise.

The fact that embedded web services are used is a strong point, since it will facilitate integration with other services and applications. By complementing the solution with the appropriate firewalling and access policies, any level of sensor access can be made possible. The implementation and proper functioning of the solution has been demonstrated through the deployment on a publicly accessible test setup. This evaluation gave us insights about the strengths and limitations of

the proposed approach in a large-scale, real-life environment.

In this work, we have also presented a novel solution for interacting with a group of CoAP resources across multiple smart objects. It provides an interesting alternative to multicast-based solutions, which are challenging to realize in a constrained environment. It is also an alternative to application-based solutions, which simply program the required functionality. An Entity Manager (EM), which can reside anywhere, turns groups of resources into entities. A strong point of our approach is that it nicely integrates the important aspects of entity management: creation, validation, usage and manipulation. At the side of the constrained devices, it requires no additional complexity, except optional support for profiles in order to realize more powerful validation. The introduction of entity profiles introduces a lot of flexibility and opportunities for further extensions regarding how entities should behave. We have implemented our proposal and demonstrated and validated its feasibility. We have also performed a detailed performance evaluation of our solution. We explored several aspects (overhead, timing, scalability, *etc*.) related to the creation, validation and usage in realistic sensor networks and compared it with existing multicast-based solutions. As such, we think that our solution is a powerful enabler for group communication in LLNs and an interesting building block for IoT applications.

In this dissertation we have tackled the problem of observing a group of CoAP resources by further exploring the potential of our unicast-based CoAP group communication approach. We have demonstrated that by relying on standard CoAP unicast messages, it becomes possible to observe CoAP groups effectively. We have also shown that several techniques can be applied in order to further reduce the number of notifications sent to the observer. This approach fits within the current spirit of distributed processing or fog computing, where processing functionality is moved closer to the data sources, as bandwidth towards the Cloud is not infinite, might be costly or might cause too high latencies.

The ability to communicate with groups of resources is important for many IoT applications. CoAP, which is expected to play an important role as an application protocol for use in constrained environment, does not have built-in group communication features. However, CoAP is designed in way that makes it easy to extend. Currently there are two main approaches to extend CoAP with group communication capabilities. The fundamental difference between the two approaches lies in the underlying communication type: multicast versus unicast. The trade-off between the two communication types is reliability versus speed. In this dissertation we proposed a hybrid solution that tries to get the benefits of both approaches. The solution is flexible, allowing the user to select the communication type based on the desired features. As such, we believe that our solution is a powerful enabler for group communication in LLNs and an interesting building block for IoT applications.

Next to this, we have experimentally evaluated those approaches using two Wireless Sensor Network (WSN) testbeds (one testbed in a shielded room and another in an office environment). We explored several aspects (overhead, timing, scalability, *etc*.) related to the usage in realistic sensor networks and compared the different approaches. Experimental evaluation reveals that there are limitations to the size of the groups. This may impact the design of real Building Automation Systems (BASs) that typically operate in conditions where interference is inevitable. A further outcome of the experimental evaluation is the impact of the CoAP parameter settings (Leisure, back-off, *etc*.) on the group communication performance.

6.2 Outlook

The work presented in this dissertation presents a solid building block for the realization of several IoT applications. However, while performing the research several opportunities for further optimizations have been identified, some of them requiring extensive research.

Security is crucial for many Internet applications. It is even more crucial for IoT applications, since now real everyday objects are exposed to Internet threats. Security breaches in the IoT can have severe impacts, sometimes even life threatening (e.g. when involving medical equipment, moving machines, *etc*.). An advantage of using unicast for CoAP group communication, rather than multicast, is that Datagram Transport Layer Security (DTLS) can be directly applied. However, the impact of using DTLS needs to be evaluated. If security solutions for multicast become available, it would be nice to compare the impact of secure communication on both communication types.

Another aspect to investigate is the combination of discovery, self-organization and resource attributes with grouping, to be able to **automatically propose and create groups**. This would allow suggesting groups to users, maybe using semantics. In addition, it will be investigated to what extend any manual configuration that is still required can be avoided and how management tools based on embedded web service technology can bring the self-organization, configuration and management of sensor networks to a next level.

During the tests on the various testbeds we observed a clear influence of protocol parameters settings on the performance, but the optimal settings depend on group and network properties. A solution might be the integration of a "cognitive loop" which **automatically and dynamically adjusts parameters settings** according to the context and on a per group basis. Also interaction with other mechanisms, such as proposed congestion control mechanisms, *etc*. is relevant. To this end we need to add more intelligence to the EM to make it optimize the response time and network overhead. One such optimization could be the adap-

tation of the CoAP retransmission parameters for certain members based on the history of communication with these members and see whether we can also **apply Quality of Service (QoS)**.

During the evaluations we have identified a number of possible paths for further optimizations and improvements. In order to improve the performance of our solutions, we like to let the EM **automatically adapt the Notifications Aggregation Window size** based on the history of the distribution of notifications.

More in depth research on different **back-off strategies and congestion control** mechanisms is needed in order to achieve an optimal balance between reliability and latency.

Also, **additional ways to interact with entities**, by extending the profiles, is an interesting research topic. Further, the application of the designed mechanisms in settings where real sensor data is being generated should be considered. This way aspects such as the impact of the correlation of data on group performance can be analyzed.

Constrained devices often make their data available using a single data format. The use of heterogeneous data formats may lead to interoperability issues. To this end, we should further extend the CoAP++ framework to **automatically interpret resource representations** (content formats), data units, possible operations, *etc*. For this we will build upon standards such as CoRE Link Format, Internet Protocol for Smart Objects (IPSO) and Open Mobile Alliance (OMA) Lightweight M2M (LWM2M). More work is needed to support automatic conversion between content formats and data units. It might also be interesting to compare the designed CoAP-based solutions with competitive protocols such as Message Queuing Telemetry Transport (MQTT).

A **tighter integration with lower layers** may further improve latency and reliability. Depending on the configuration of the groups and the type of resources involved, application requirements can be derived and translated into optimal configuration of the lower layers – e.g. optimized Media Access Control (MAC) schedule, Time Division Multiple Access (TDMA), IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH), local retransmissions, *etc*. This way, powerful IoT application enablers at the higher layers can be combined with advancements at the lower layers, together delivering the performance expected by IoT applications and their users.

A

IETF Standardization in the Field of the Internet of Things (IoT): A Survey

In this appendix, we survey IETF standardization efforts in the field of the IoT. Hereby, we consider the complete stack starting with 6LoWPAN and ending with CoAP. This standardized stack is also the basis this PhD research builds upon.

⋆ ⋆ ⋆

I. Ishaq, D. Carels, G. Teklemariam, J. Hoebeke, F. Van den Abeele, E. De Poorter, I. Moerman, and P. Demeester.

Published in the Journal of Sensor and Actuator Networks, 25 April 2013.

Abstract Smart embedded objects will become an important part of what is called the Internet of Things. However, the integration of embedded devices into the Internet introduces several challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. In the past few years, there have been many efforts to enable the extension of Internet technologies to constrained devices. Initially, this resulted in proprietary protocols and architectures. Later, the integration of constrained devices into the Internet was embraced by IETF, moving towards standardized IP-based protocols. In this paper, we will briefly review the history of integrating constrained devices into the Internet, followed by an extensive overview of IETF standardization work in the

6LoWPAN, ROLL and CoRE working groups. This is complemented with a broad overview of related research results that illustrate how this work can be extended or used to tackle other problems and with a discussion on open issues and challenges. As such the aim of this paper is twofold: apart from giving readers solid insights in IETF standardization work on the Internet of Things, it also aims to encourage readers to further explore the world of Internet-connected objects, pointing to future research opportunities.

A.1 Introduction

Internet protocol technology is rapidly spreading to new domains where constrained embedded devices such as sensors and actuators play a prominent role. This expansion of the Internet is comparable in scale to the spread of the Internet in the '90s and the resulting Internet is now commonly referred to as the Internet of Things (IoT). The integration of embedded devices into the Internet introduces several new challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. These embedded devices are typically designed for low cost and power consumption and thus have very limited power, memory and processing resources and are often disabled for long-times (sleep periods) to save energy. The networks formed by these embedded devices also have different characteristics than those typical in today's Internet. These constrained networks have different traffic patterns, high packet loss, low throughput, frequent topology changes and small useful payload sizes.

In the past few years, several innovations were developed to enable the extension of Internet technologies to constrained devices, moving away from proprietary architectures and protocols. Most of these efforts focused on the networking layer: *IPv6 over Low-Power Wireless Personal Area Networks* (RFC 4919) [1], *Transmission of IPv6 Packets over IEEE 802.15.4 Networks* (RFC 4944) [2], IETF *routing over low-power and lossy networks* [3] or the ZigBee adoption of *Internet Protocol Version 6* (IPv6) [4]. These new standards enable the realization of an Internet of Things, where end-to-end IP-based network connectivity with tiny objects such as sensors and actuators becomes possible.

However, it was not global connectivity that was at the basis of the great success of the current Internet, but the World Wide Web and the resulting web service technologies. Today, an embedded counterpart of web service technology is needed in order to exploit all great opportunities offered by the Internet of Things and turn it into a Web of Things. Recently, standardization work has started within the IETF *Constrained RESTful Environments* (CoRE) working group [5] to allow the integration of constrained devices with the Internet at the service level.

With these technologies, it has now become possible to deploy a self-organizing sensor network, to interconnect it with IPv6 Internet and to build applications that

interact with these networks using embedded web service technology. Application developers using high-level programming languages can count on these standardized technologies in the realization of a Semantic Web of Things or Sensor Web, which enables data producers and users to publish and access sensor information via web- and standards based interfaces. In this paper, we will briefly review the history of integrating constrained devices into the Internet, with a prime focus on the IETF standardization work. The key realizations of the related IETF working groups will be discussed, complemented with an extensive overview of related research results that illustrate how these novel technologies can be extended or used to tackle other problems and with a discussion on open issues and challenges.

The remainder of this paper is organized as follows. In Section A.2, we discuss the evolution from proprietary solutions towards IP-based integration of constrained devices using standardized protocols, introducing the most relevant standardization bodies and groups. In the following sections we will extensively discuss these standardization efforts, related work and open issues and challenges. Section A.3 introduces IEEE 802.15.4 as it is the most widely used physical layer and MAC layer in constrained networks, providing the foundations for the networking protocols at the higher layers. From that point on, the focus is shifted to the IETF standardization, discussing IETF 6LoWPAN, IETF RoLL and IETF CoRE extensively in Sections A.4, A.5 and A.6 respectively. Section A.7 glues everything together, illustrating how all these standardization efforts contribute to the realization of the Internet or Web of Things. Finally, section A.8 concludes this paper.

A.2 Integration of constrained devices into the Internet

In the absence of widely accepted standard protocols for resource-constrained devices, out-of-necessity many vendors developed proprietary protocols to run inside their sensor networks. Connectivity between the Internet and the sensor networks was achieved through the use of vendor-specific gateways or proxies. These gateways have to translate between protocols used in the Internet and proprietary protocols used in the sensor networks. Figure A.1 displays two different sensor networks that are connected to the Internet by gateways. Users on the Internet have to connect to the gateways in order to obtain data from the corresponding sensor network. There are several ways how a gateway can handle such user requests. For example, the gateway from vendor 1 translates standard Internet protocols into proprietary sensor protocols and relays the requests to the sensors in its network. The gateway then receives the answers from the relevant sensors by means of the proprietary sensor protocols and sends back the appropriate reply to the user using

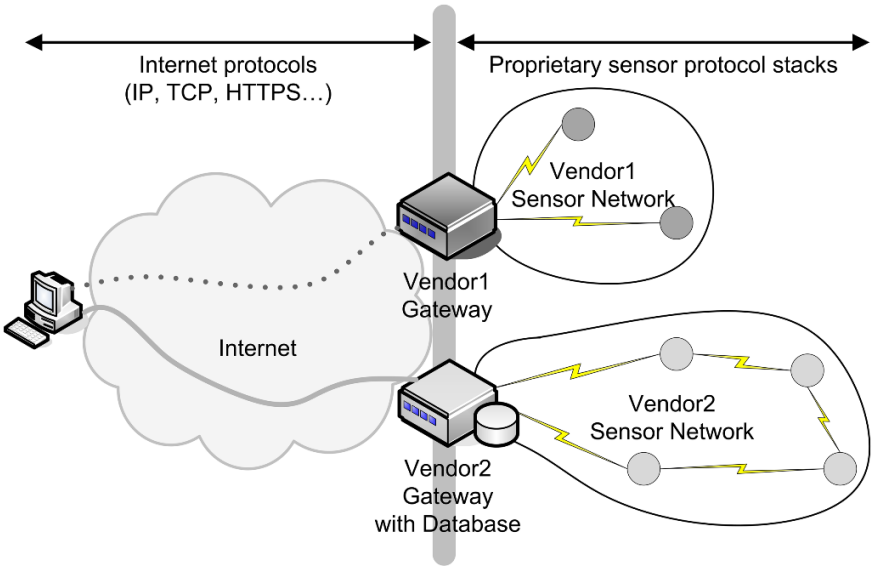


Figure A.1: Gateways and proprietary protocols are often used to interconnect sensor networks to the Internet.

standard Internet protocols. The gateway offers an API that applications should use in order to create requests that can be understood by the gateway. This approach has the benefit that direct (real-time) interaction with sensor nodes is possible, but only by using a vendor-specific interface. Alternatively, the gateway of vendor 2 contains a database with pre-collected sensor data. When it gets a request from a user on the Internet, it replies directly to the requester using the data in the database. In some cases, the gateway is simply running a web server that makes the data available to the outside world. In this case, existing database technologies can be reused, but the user does not know whether the returned data is coming in real-time from the sensors or whether it is coming from a value that has been previously stored in a database.

The use of standardized solutions is mostly limited to the use of a standard for the physical layer and MAC layer, although tailored MAC protocols could be used as well. It is clear that such an approach hinders the integration of sensors into the Internet. Little flexibility is offered since users can query the sensors only in the way that is allowed by the gateway. Another disadvantage is the vendor lock-in: gateways and sensors often have to be from the same vendor in order to be compatible. In addition, creating and maintaining the gateway requires significant development effort: often even adding new sensor resources requires making (administrative) changes to the gateway. Finally, due to the lack of real end-to-end connectivity, no real-time interaction with the resource-constrained devices is

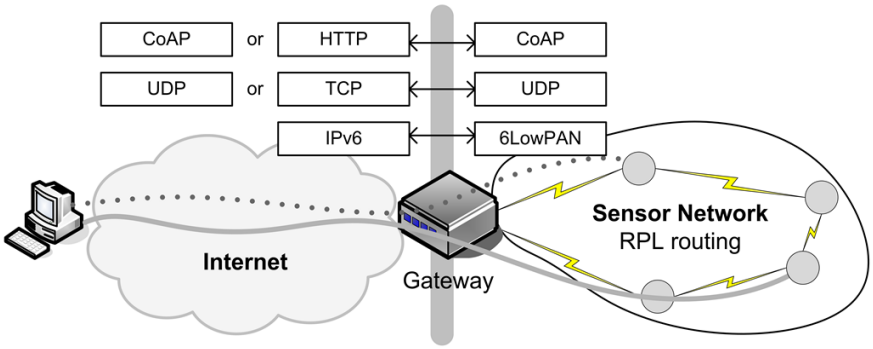


Figure A.2: Internet protocols are extended to the sensor networks. The Gateway translates between the two standardized protocol stacks.

supported.

These limitations in combination with the general understanding that constrained devices will take up a prominent role in the future Internet, raised the need for standardized, open solutions for network communication with constrained devices. To ensure wide adoption, these new solutions have to be interoperable with the most widely used protocols in the Internet, initially IP and, in a later stage, HTTP. To address these needs, the IETF—responsible for the development of high-quality Internet standards—has formed several working groups: *IPv6 over Low Power WPAN* (6LoWPAN) [6], *Routing Over Low Power and Lossy Networks* (ROLL) [3] and *Constrained Restful Environments* (CORE) [5]. The 6LoWPAN group tackles the transmission of IPv6 packets over IEEE 802.15.4 networks, the ROLL group develops IPv6 routing solutions for Low Power and Lossy Networks (LLNs) and the CoRE group aims at providing a framework for resource-oriented applications intended to run on constrained IP networks. Together, these protocols allow the IP-based integration of constrained devices into the Internet in a standardized way, as shown in Figure A.2. Due to the popularity of IEEE 802.15.4, this standard is used at the physical layer and the medium access control layer.

Similar to the previous approaches, gateways are still used to translate between the protocols used in the Internet and protocols used in the sensor networks, e.g., IPv6 to 6LoWPAN and vice versa. However, due to the use of standardized protocols, many of the disadvantages from the previous approaches are now solved. For example it is now possible to combine sensor devices from different vendors in the same network, or to use a gateway from a different vendor than the vendor of the sensor devices. Flexibility is also improved by this approach as users are not confined to the API offered by the gateway: users can directly query the sensors without the need for the gateway that understands the query or needs to interpret the data. The application payload can now travel directly from the client

to the sensor, where it is processed and acted upon. The gateway takes care of the translation between standardized protocols. This end-to-end approach makes adding and removing sensor resources transparent to the gateway and improves interoperability of devices.

A.3 IEEE 802.15.4

To create a standardized protocol stack for constrained networks and devices, the IETF builds further upon the IEEE 802.15.4 standard. The IEEE 802.15.4 standard is maintained by the IEEE 802.15 working group [7] and defines low-data-rate, low-power, and short-range radio frequency transmissions for *wireless personal area networks* (WPANs). The working group aims to keep the complexity of the standard and the cost of the necessary hardware low, making it suitable for wireless communication among constrained devices such as sensors and actuators. The standard describes a Physical (PHY) layer and a *Medium Access Control* (MAC) sublayer. These will be discussed in the following two subsections. Furthermore, 802.15.4 has proven to be a popular technology for wireless communication in WPANs and a number of examples that have adopted 802.15.4 are listed in the last subsection.

A.3.1 Physical Layer

The IEEE 802.15.4 physical layer is responsible for (de)activating the radio transceiver, data reception and transmission, channel frequency selection, energy detection within a channel and determining whether or not the communication channel is occupied by a transmission (*i.e.*, Clear Channel Assessment, CCA). IEEE Std 802.15.4–2011 [8] defines a total of 15 different PHY modes. The PHY mode specifies which frequency band and modulation are used for data transmission.

An 802.15.4 compliant radio typically operates at one of the following license-free frequency bands: 868–868.6 MHz (e.g., Europe), 902–928 MHz (e.g., North America) or 2,400–2,483.5 MHz (worldwide, *i.e.*, ISM band). O-QPSK and BPSK are the most commonly used constellations for (de)modulating the signal. Almost all of the defined PHY modes apply a spreading technique that allows spreading out the signal so that it occupies a larger bandwidth. By using these simple constellations and the spreading technique, the communication is more robust and has an increased resistance to interference and (narrow-band) noise even when using a low transmission power.

Depending on the environment and the PHY mode, the communication range varies between 10 and 100 meters. In 802.15.4–2011 the maximum PHY data rate for the ISM band is limited to 250 kbps when using *Direct-Sequence Spread Spectrum* (DSSS) or 1,000 kbps when using *Chirp Spread Spectrum* (CSS) respectively

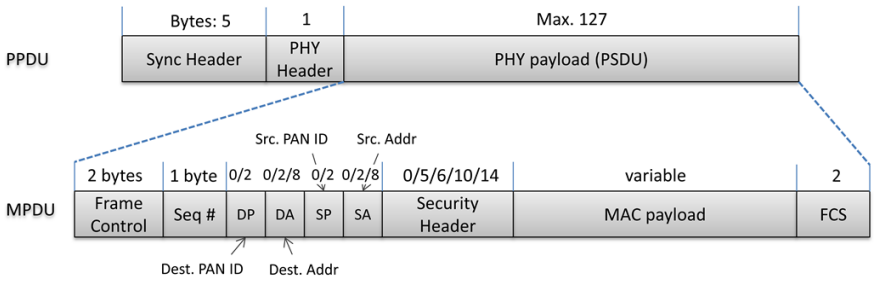


Figure A.3: IEEE 802.15.4 The Physical Protocol Data Unit (PPDU) and MAC Protocol Data Unit (MPDU) formats.

as a spreading technique. CSS is, however, an optional feature of the standard and in most applications 250 kbps is the maximum PHY data rate.

A.3.2 MAC Sublayer

Because devices that are in each other's proximity share the same communication medium, the access to the medium has to be controlled. When a device wants to send a packet, the MAC sublayer asks the PHY layer's CCA to check whether the medium is occupied. If the CCA indicates that another transmission is currently taking place, the MAC sublayer defers its transmission and waits for a set amount of time before it retries sending the packet. If, on the other hand, the CCA determines the medium to be free then the MAC sublayer transmits the packet immediately.

Apart from medium access control the MAC sublayer also provides acknowledgement of frame reception and validation of incoming frames. The standard defines three different network structures: star topology, mesh topology and cluster tree topology. However, these 802.15.4 topologies are hardly ever used in practice and protocols that run on top of 802.15.4 build their own networks instead. To limit energy consumption, duty cycles are often used by MAC protocols. As a result, transceivers can be in sleeping mode up to 99% of the time, which drastically reduces power consumption and increases operational lifespan.

The *Physical Protocol Data Unit* (PPDU) and *MAC Protocol Data Unit* (MPDU) formats as defined in IEEE Std 802.15.4–2011 are shown in Figure A.3. The PPDU for O-QPSK consists of a SYNC header (for receiver clock synchronization), a PHY header (which contains the length of the PSDU) and a PHY payload referred to as the *Physical Service Data Unit* (PSDU). To limit packet error rates, the PSDU is limited in size to 127 bytes. An 802.15.4 MPDU or MAC frame is transported as the PSDU of the PPDU and consists of three parts: (i) the *MAC header* (MHR), (ii) the MAC payload and (iii) the *MAC footer* (MFR). The MHR

comprises frame control (indicating the type of frame), sequence number (for referring to frames, e.g., in acknowledgements), addressing information (for source and destination identification) and security-related information, and is of variable length. The frame payload also has a variable length and contains information specific to the frame type. Finally, the MFR is 2 bytes long and contains a *frame check sequence* (FCS) that is calculated by the sender and that is used for frame validation by the receiver.

Depending on the frame type the length of the addressing fields varies. An acknowledgement control frame for instance will not contain any addressing information. For data frames the length depends on which addressing format is being used and whether or not (optional) PAN identifiers of 2 bytes are included. The standard specifies two addressing formats: long 64-bit addresses that are globally unique and short 16-bit addresses that are unique within a PAN. When using the long addressing format, the MHR's length is 19 bytes and the maximum size of the payload is limited to 106 bytes. For the short addressing format the MHR is 7 bytes long and the maximum payload size is limited to 118 bytes. Using link-layer security can further reduce the size available for the payload size by as much as 14 bytes.

A.3.3 IEEE 802.15.4 based solutions

Several higher layer protocols have been defined directly on top of the IEEE 802.15.4 MAC sublayer. For completeness, a very brief overview of the most commonly used non-IETF solutions that have been built on top of IEEE 802.15.4 is given below.

ZigBee [9] builds upon the physical layer and medium access control defined in IEEE standard 802.15.4 for low-rate WPAN with additional network, security and application software layers. Predefined application services specify which actions a device can take, the main example being "turn the lights on" and "turn the lights off".

- Wireless HART [10] focuses on automation and industrial applications that require real time guarantees. To realize these goals, a time synchronized, self-organizing, and self-healing mesh architecture is used. The standard was initiated in early 2004 and developed by 37 *HART Communications Foundation* (HCF) companies. In April 2010, WirelessHart was approved by the *International Electrotechnical Commission* (IEC) unanimously, making it a wireless international standard as IEC 62591.

- The MiWi protocol stacks [11] are small foot-print alternatives to ZigBee (40K–100K), which makes them useful for cost-sensitive applications with limited memory. Although the MiWi software is free, there exists a unique restriction and obligation to use it only with Microchip microcontrollers.

- ISA100 [12] addresses wireless manufacturing and control systems (developed by the *Systems and Automation Society* (ISA)). They defined ISA100.11a, a wireless networking standard that builds upon IEEE 802.15.4.

Contrary to most of the above solutions, the IETF standards are fully open (no contributor fees are required). To ensure that each device is IP-addressable, an IPv6 layer (6LoWPAN layer) is defined above the IEEE 802.15.4 MAC sublayer. This adaptation layer is discussed in the next section.

A.4 IETF 6LoWPAN Working Group (IPv6)

The IPv6 protocol has a high overhead and restrictions that make it unsuitable for LLNs such as IEEE 802.15.4 networks. For instance, considering the limited space available for the MAC payload in an 802.15.4 MPDU, the use of a 40-byte IPv6 header would be too excessive. Therefore, the IETF 6LoWPAN (IPv6 for Low-power Wireless Personal Area Network) Working Group was formed to work on the IPv6 protocol extensions required for such networks where the nodes are interconnected by IEEE 802.15.4 radios [13]. Meanwhile the working group has produced several proposed standards and informational documents regarding these required extensions.

Starting from a well-defined set of assumptions and a problem statement, as defined in the informational RFC 4919 [1], a solution for transmitting IPv6 packets over IEEE 802.15.4 networks was defined, resulting in RFC 4944 [2]: *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. This document describes the frame format, the methods of link-local address formation and autoconfigured addresses, simple header compression and mesh-under routing for multi-hop IEEE 802.15.4 networks. Subsequent RFCs of the 6LoWPAN working group, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks* (RFC 6282) [14] and *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)* (RFC 6775) [15], which respectively cover advanced header compression and Neighbor Discovery optimization, have updated RFC 4944. In addition to this, the working group has produced 2 other informational RFCs addressing use case descriptions (RFC 6568) [16] and routing requirements (RFC 6606) [17] and is in the process of finalizing an internet draft on the transmission of IPv6 packets over Bluetooth (draft-ietf-6lowpan-btle-11) [18].

Further, the working group is also expected to collaborate with other organizations (such as IEEE and ISA SP100) and other IETF working groups (such as ROLL) on common interest issues and is mandated with providing implementation and interoperability guides [19].

A.4.1 Key protocols

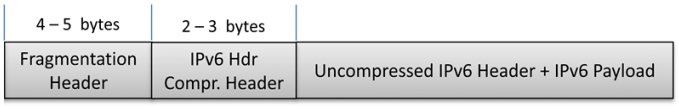
As mentioned in Section A.4, the main objective of the 6LoWPAN working group is coming up with extensions to IPv6 protocols so that IPv6 packets can be transferred in constrained networks such as IEEE 802.15.4. IPv6 forwarding routers, unlike IPv4 routers, do not support fragmenting outgoing packets. This means that the communicating hosts have to send packets with the right Maximum Transmission Unit (MTU) supported by the communication links of the router. To meet this requirement hosts may use Path MTU Discovery to find a suitable MTU along the path or just send packets that meet the minimum MTU requirement for all links, which is 1,280 bytes. However, this minimum value is still too big for IEEE 802.15.4 links that have an MTU of 127 bytes for the entire MPDU. To use IPv6 on top of IEEE 802.15.4 networks, a mechanism that allows transmitting IPv6 packets that are larger than the MTU of 127 bytes is required. The solution presented by the 6LoWPAN working group is to use a layer between the network and the data link layers that supports packet fragmentation and reassembly. In addition, the 40 byte IPv6 fixed header takes a significant portion of the already small protocol data unit of the LLNs, leaving little room for IPv6 data payload. To solve this problem different sorts of header compression are proposed by the working group. Further, these fragments have to be routed between the Low-power and Lossy Network (LLN) nodes. Layer 2 multi-hop data transmissions should also be addressed in relation to IPv6 adaptation. Accordingly, the 6LoWPAN working group has introduced an IPv6 adaptation layer, named 6LoWPAN Adaptation Layer that lies between the data link layer and the network layer of the protocol stack. The adaptation layer delivers three basic services: packet fragmentation and reassembly, header compression, and data link layer routing (for multi-hop connections).

A.4.1.1 6LoWPAN frames

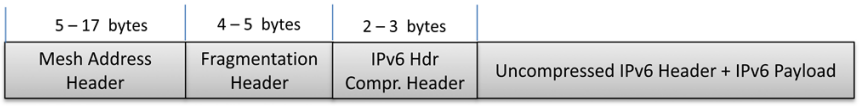
RFC 4944 states that all LoWPAN encapsulated datagrams are prefixed with an encapsulation header stack where each header in the header stack contains a header type and zero or more header fields. The LoWPAN header may contain the mesh addressing header, fragmentation header, and IPv6 Header compression header, in that order. Figure A.4 shows examples of header stacks that might be used in 6LoWPAN. These frames are transported inside the frame payload field of an 802.15.4 data MPDU.

Similar to IPv6, the 6LoWPAN headers are added when required. The first byte of the encapsulation header, the dispatch byte, identifies the next header. More specifically, the first three bits of the dispatch byte indicate the next header type while the remaining bits are used for different purposes depending on the header type. Table 1 summarizes the different values for the dispatch byte.

(a) 6LoWPAN encapsulation header stack with IPv6 header compression header.



(b) 6LoWPAN encapsulation header stack containing fragmentation and IPv6 header compression header.



(c) 6LoWPAN encapsulation header stack containing mesh addressing header, fragmentation and IPv6 header compression header.

Figure A.4: 6LoWPAN encapsulation header stack examples.

Table A.1: Summary of Dispatch Byte values.

First 3 Bits	**Header Type**	**Description**
00x	NLAP	This is not a 6LoWPAN frame. This is important for 6LoWPAN to co-exist with other protocols. The remaining 6 bits are ignored.
010	Uncompressed/HC1 Compressed IPv6 Addressing Header	The address type is determined depending on the remaining 5 bits. E.g.: 00001 = uncompressed IPv6 Address 00010 = HC1 Compressed Header.
011	IPHC Compressed Header	The remaining 5 bits are added to the rest of IPHC compression header to optimize IPv6 header compression.
10x	Mesh Header	The next header is the mesh header. The last bits are used for other purposes related to mesh-under routing.
11x	Fragmentation Header	The next header is a fragment header. The fragment type is determined by the remaining 6 bits. The bit sequence 000xxx indicates first fragment while 100xxx indicates Non-first fragments. The last three bits in both types of fragments will be used for other purposes. The other bit sequences are reserved.

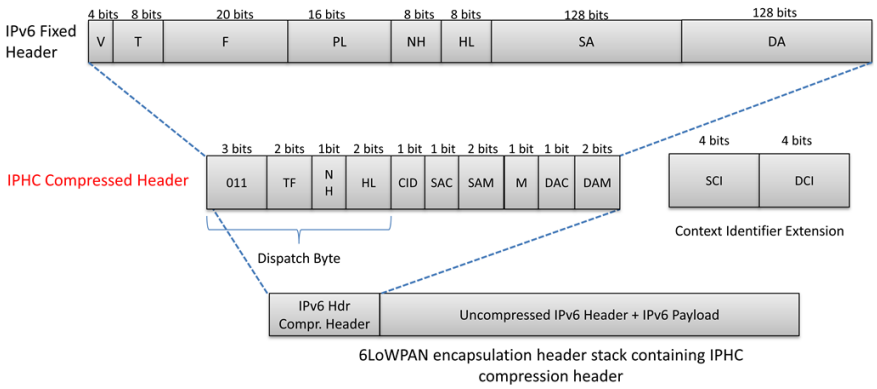


Figure A.5: IPHC Compression.

A.4.1.2 Header compression

One of the services provided by the 6LoWPAN adaptation layer is header compression. RFC 4944 defines header compression techniques that can be used to compress IPv6 headers and to allow more data from layers on top of IPv6 to be included in a single 802.15.4 frame.

The LOWPAN_HC1, the main compression technique specified in RFC 4944, is optimized for compressing IPv6 packets that contain link-local IPv6 addresses. The technique attempts to reduce the size of the packet by removing common fields (Version, TC, Flow label), inferring the IPv6 addresses from the link-layer addresses in the 802.15.4 header and the static IPv6 link-local prefix (fe80::/64), inferring the IP packet length from the layer 2 frame length (or the fragmentation header) and limiting the values of the next header field to TCP, UDP and ICMP [20]. The RFC also defines HC2 compression for transport layer compression, which allows compressing UDP, TCP and ICMP. Other transport layer protocols cannot be compressed by HC2. HC1 compression has very low compression factors for global and multicast addresses, which are needed for direct host-to-host interactions between constrained devices and clients as envisioned by the Internet of Things, therefore its use is limited and not further detailed here.

RFC 6282 specifies two new compression mechanisms named LOWPAN_IPHC and LOWPAN_NHC. According to the RFC, LOWPAN_IPHC, uses 13 bits for compression (the last 5 bits of the dispatch byte and an additional byte) and an extra 8 bits to store context information, when necessary. Figure A.5 shows the IPHC header format and Table A.2 summarizes the address compression fields.

IPHC is based on a number of basic assumptions about common 6LoWPAN communication cases. The first assumption regards commonly used fields. It is assumed that IPv6 header fields such as Version, Traffic Class and Flow Label have

Table A.2: Summary of address compression fields.

Field	Description
Context ID Extension (CID) (1 bit)	0 = No Additional Context Identifier Extension is used. 1 = An additional 8-bit field follows the DAM field
Source Address Compression (SAC) (1 bit) Dest. Address Compression (DAC) (1 bit)	0 = Stateless source/destination address compression 1 = Stateful, context based source/destination address compression
Source Address Mode (SAM) If SAC = 0	00 = The full 128 bits address is sent inline 01 = The last 64 bits are sent inline 10 = The last 16 bits are sent inline. 11 = The entire source address is elided
If SAC = 1	00 = The unspecified address, :: . Nothing is sent inline 01 = 64 bits are carried inline 10 = 16 bits are carried inline 11 = The address is fully elided
Multicast Compression (M)	0 = Destination address is not a multicast address 1 = Destination address is a multicast address
Destination Address Mode (DAM) If M = 0 and DAC = 0	Same as SAM with SAC = 0
If M = 0 and DAC = 1	Same as SAM with SAC = 1
If M = 1 and DAC = 0	00 = The full address is sent inline 01 = Only 48 bits are sent inline 10 = Only 32 bits are sent inline 11 = Only 8bits are sent inline
If M = 1 and DAC = 1	00 = Only 48 bit s are sent inline. 01, 10, 11 = Reserved

fixed values and do not have to be transmitted. Therefore, IPHC totally ignores the 4 bit version and attempts to compress the Traffic Class and Flow Label into the TF bits (2 bits). By assuming hop limits will be set to well-known values such as 1, 64 and 255 by the host, IPHC compresses the hop limit from 8 bits to 2 bits.

Assuming that addresses could be generated from link-layer addresses and that mostly link local addresses are used inside LLNs, IPHC attempts to compress the IPv6 address fields. To further improve the efficiency of compression of global and multicast addresses, IPHC uses context information. The *Context ID* Extension bit (CID) bit indicates whether an additional 8-bit Context Identifier Extension field is added or not. IPHC supports stateless and stateful methods of addresses compression. The *Source Address Compression* (SAC) and *Destination Address Compression* (DAC) bits indicate which of these two compression methods is used for the source and destination address, respectively. The *Source Address Mode* (SAM) bits in combination with the SAC bit determine how many of the source address

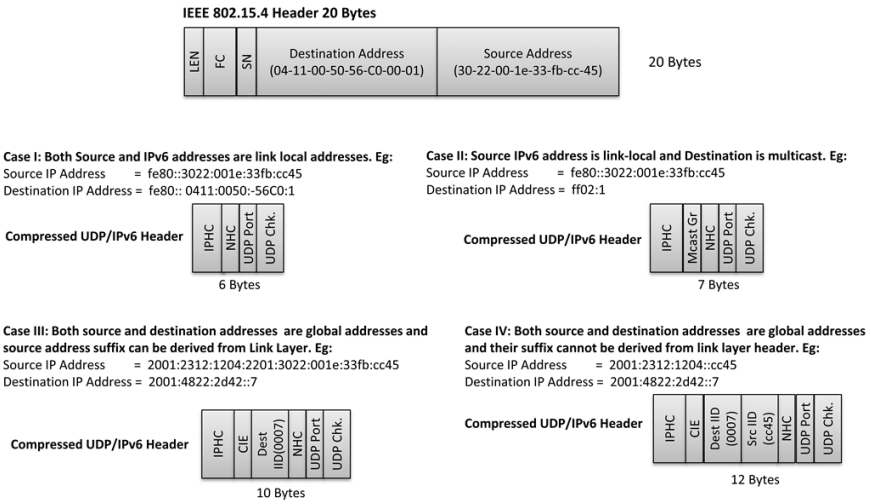


Figure A.6: Internet Protocol Version 6 (IPv6) Header Compression Example.

bits are actually elided and how many bits are sent in-line. The number of bits sent inline can be 128, 64, 16 or 0 bits for stateless compression and 64, 16, or 0 bits for stateful compression. In all cases, the bits that are elided are assumed to be calculated from the link-local prefix, link layer address and stored context. The destination address compression varies based on the address type. Multicast destination addresses are indicated by the M bit in the compression header. Unicast destination addresses are compressed in the same way as unicast source addresses. However, multicast addresses follow a different rule based on the DAC bit. Accordingly, 128, 48, 32, or 8 bits has to be sent inline when multicast destination addresses are compressed. The 8 bit context identifier is added only if the SAC and/or DAC bits indicate its presence. If it exists, the first 4 bits indicate store context for the source address while the remaining bits store destination address context. The RFC does not specify what is stored in these fields nor how communicating parties exchange this information.

The IPv6 Payload Length field can be inferred from the fragment header or from the link layer header and has to be fully elided.

As was already mentioned HC2 can only compress UDP, TCP and ICMPv6 headers. To alleviate this issue LOWPAN_NHC introduces a variable length next header identifier which could be used for future next header compressions.

Figure A.6 illustrates the IPHC header compression when link-local, global and multicast IPv6 addresses are used for communication.

In case I, since both source and destination IP addresses are link-local unicast addresses, the prefix is fixed (*i.e.*, fe80::/64) and the suffix can be inferred from the IEEE 802.15.4 source and destination addresses. As shown in the figure, the entire

IPv6/UDP header can be compressed from 48 bytes to just 6 bytes. The second case illustrates IPHC compression when the destination IP address is a multicast address. Here the IPv6/UDP header is compressed to 7 bytes by sending only the multicast group inline and deriving all other information from the IEEE 802.15.4 header. Cases III and IV are typical to IoT interactions where both source and destination IP addresses are global unicast addresses. In case III, the source suffix can be derived from the IEEE 802.15.4 source address and, hence, does not have to be sent in-line. The destination suffix however cannot be derived from the IEEE 802.15.4 destination address and has to be sent uncompressed. In case IV, the suffixes of both source and destination addresses cannot be derived and have to be sent inline. In cases III and IV, the IPv6 address prefixes and other information is derived based on the shared context information that is stored in the *Context Identifier Extension* (CIE) byte. In case 3 and case 4, the IPv6/UDP headers are compressed from 48 bytes to 10 and 12 bytes respectively.

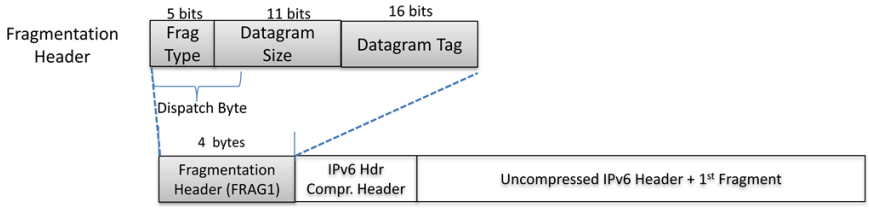
A.4.1.3 Fragmentation

The other service provided by the 6LoWPAN adaptation layer is fragmentation and reassembly. Fragmentation is only required when the entire IPv6 packet cannot fit in a single IEEE 802.15.4 frame, *i.e.*, when it is larger than the available space for the MPDU payload (typically 106 bytes, see Section A.3.2). According to RFC 4944, fragmentation breaks a single IPv6 packet into smaller pieces and a fragmentation header is included in every fragment. The document further specifies using two different types of fragmentation headers. The fragment header of the first fragment contains only the datagram size (11 bits) and datagram-tag (16 bits) fields, while subsequent fragments of the same IPv6 packet also include the datagram-offset (8 bits) field. Datagram size is the length of the entire unfragmented IPv6 packet, datagram-tag identifies to which datagram (*i.e.*, packet) a particular fragment belongs. Therefore, these values have to remain the same for all fragments of a single IPv6 packet. The datagram-offset indicates the offset of the fragment from the first fragment. Figure A.7 shows the 6LoWPAN fragmentation header.

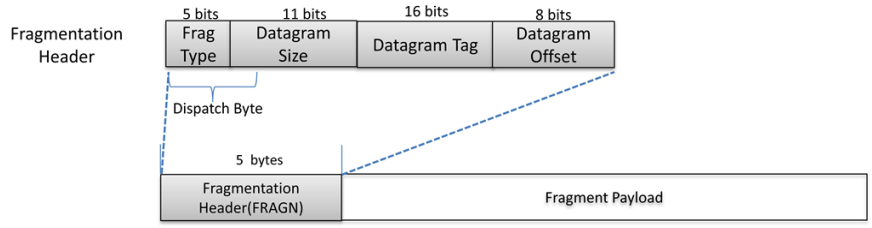
A.4.1.4 Mesh-Under Routing support

Routing involves calculating best paths (according to some metric) to a destination in a multi-hop network. If a single link layer technology (such as IEEE 802.15.4) is used at layer 2, the path calculation could be done at either layer 3 or layer 2. Layer 3 routing is usually referred to as route-over routing while forwarding at layer 2 is called mesh-under routing.

Layer 3 routers build and maintain a routing-table which contains the next-hop to all known destination networks based on their network prefixes. This means



(a) 6LowPAN encapsulation header stack for the first fragment, containing the Fragmentation header (FRAG1) and IPv6 Header Compression header.



(b) 6LowPAN encapsulation header stack for subsequent fragments, containing the Fragmentation header (FRAGN).

Figure A.7: 6LoWPAN Fragmentation Header.

that when a packet arrives at a router, the link-layer encapsulation is removed and the destination address in the IPv6 header is used to do a longest prefix match in the routing table to determine the next hop. Once the next hop router is known, the router re-encapsulates the packet with layer 2 headers and trailers. The layer 2 addresses indicate the current router as a source and the next hop router as the destination while the source and destination IP addresses remain unchanged. This is done at every forwarding router along the path of the packet.

Mesh-under routing on the other hand, uses link layer addresses to make forwarding decisions. Every forwarding layer 2 router along the path of a packet is expected to maintain its own forwarding-table based on link layer addresses in order to make forwarding decisions based on these addresses. Just as in route-over routing, four addresses are required to forward the packet at an intermediate node—the originator address, the final destination address, the current forwarding router address and the next hop router address. The IEEE 802.15.4 source and destination MAC addresses of the frame indicates the current forwarding router and the next-hop router, respectively. A way of transmitting the originator and final destination addresses is required to fully support mesh-under routing.

RFC 4944 introduces the Mesh Address Header (Figure A.8) for this purpose. The mesh address header contains the dispatch byte and the originator and final destination link layer addresses. The OA and FD bits in the header indicate which of the two 802.15.4 addressing formats are used for the Originator and Final Des-

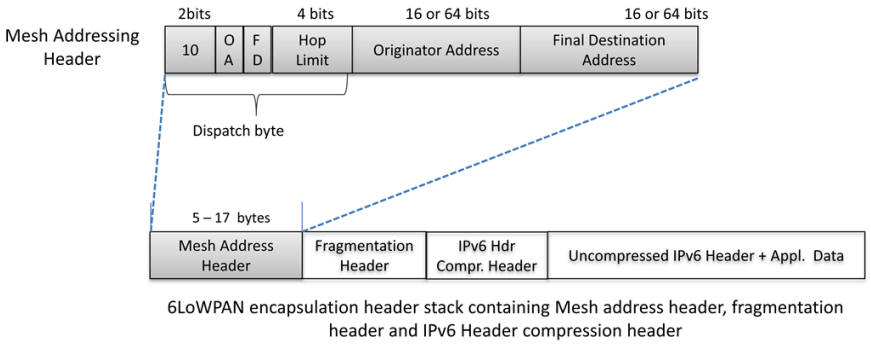


Figure A.8: 6LoWPAN Mesh Addressing Header.

tination address fields: short 16-bit or long 64-bit addresses. The hop count takes up an additional 4 bits, supporting up to 14 hops. The value 0xF is reserved to indicate that one more byte follows to support more than 14 hops.

As an example of mesh-under routing, assume that node A sends a packet to node B which is located some hops away from node A in an IEEE 802.15.4 network. First, node A, construct a mesh addressing header with its own link-layer address as Originator address and the link-layer address of node B as Final Destination address. Based on its layer 2 "routing-table", node A identifies its next hop router (node C). Accordingly, it puts node C's link layer address as the destination address and its own link-layer address as the source address in the 802.15.4 Mac Header. The resulting frame contains the following addresses in the 6LoWPAN mesh addressing header: Originator address (node A), final destination address (node B) and in the IEEE 802.15.4 MHR: Source address (node A) and destination address (node C). Once the router, node C, receives the frame, it checks the Mesh Addressing Header to check whether it is the final destination. Since node B is the final destination, node C will also forward the packet to its next hop. After determining the next hop, node C re-encapsulates the frame with its own link layer address as the source and the next hop's link layer address as the destination address in the MHR. This process continues until the packet reaches node B. The addresses in the 6LoWPAN Mesh Addressing Header remain the same along the path of the packet.

A.4.2 Implementation and evaluation

A.4.2.1 Implementation

Many organizations and companies are incorporating the 6LoWPAN adaptation layer in their protocol stacks. Additionally, there are also a couple of network simulators that support 6LoWPAN. Table A.3 summarizes the features of six different

Table A.3: 6LoWPAN implementations.

Implementation	Operating System /Simulator	License	RFC 4944	RFC 6282	RFC 6775
SICSLOWPAN	ContikiOS/Cooja Simulator	Open Source	X	X	X
BLIP (Berkley Low-power IP)	TinyOS	Open Source	X		
Arch Rock 6LoWPAN	TinyOS	Open Source	X		
NanoStack 6lowpan	FreeRTOS	Open Source	X	X	X
Hitachi	-	Commercial	X		
NS-3	Simulator	Open Source	X		

implementations of 6LoWPAN.

A.4.2.2 Evaluation

Very few performance evaluations of the 6LoWPAN adaptation layer have been published thus far. The most notable performance evaluation of the 6LoWPAN protocol was made by [21] in which the author qualitatively and quantitatively explains the advantages of 6LoWPAN as compared to different industry standards. According to [21], in terms of memory footprint, ease of deployment, scalability, energy efficiency and other features, 6LoWPAN has significant advantage over other standards, such as Zigbee.

When analyzing 6LoWPAN in detail, it can be seen that the adaptation layer solves many of the issue relating to supporting IPv6 on constrained devices and LLNs. One of the functions of the adaptation layer is fragmentation and reassembly. In order to fit large IPv6 packets into the small MTU of IEEE 802.15.4 networks and improve efficiency of communication, packets must be fragmented at the adaptation layer before it is passed on to the 802.15.4's MAC sublayer. Similarly, during reception, these fragments should be reassembled at the same layer before passing them onto upper layer protocols. This approach has enabled smart objects to participate in any IPv6 based communication with some extra processing overhead. In [22], the authors attempted to test performance of 6LoWPAN on TelosB and MikaZ motes. The authors tried to measure the impact of packet size on RTT and packet loss by increasing the packet size. As expected, the RTT increased for larger packets due to fragmentation and reassembly of packets.

Both route-over and mesh-under routing mechanisms can be used in 6LoWPAN networks. Due to fragmentation and header compression at the 6LoWPAN adaptation layer, packets in a network that uses route-over routing have to be reassembled at every intermediate node to get the destination IP address for de-

termining the next hop. These fragments have to go through compression and fragmentation processes before they are forwarded to the next hop router. This approach requires extra buffer space to store all fragments of a particular packet at each intermediate node. As memory is one of the constrained resources, this may not be an optimal solution for Internet of Things nodes. On the other hand, mesh-under routing may solve the issue of reassembly and fragmentation by routing the fragments independently. To the best of our knowledge, there is no performance comparison of the two routing approaches in literature. Also, no standardized solution for mesh-under routing has been proposed; as opposed to the RPL route-over routing protocol that is being defined in the IETF ROLL working group. However, examples of mesh-under routing can be found in literature, e.g., in [23].

IPv6 headers have a minimal length of 40 bytes. This is a too large overhead for the 127 byte MTU of IEEE 802.15.4 networks. 6LoWPAN uses different mechanisms to compress IPv6 headers by removing redundant information from the headers and inferring values from the link layer information. The original compression mechanism proposed in RFC 4944, called HC1, compresses source and destination addresses by assuming that they are link local addresses. If addresses are global, they have to be sent uncompressed. This makes HC1 compression less efficient for IoT solutions which mostly involve global IPv6 addresses. In addition to this, the transport layer protocol compression mechanism, HC2, compresses only limited numbers of protocols. New compression mechanisms, called IPHC and NHC, were introduced in RFC 6282. These mechanisms are significant improvements to the HC1 and HC2 compression mechanisms and new implementations are urged to use the new compression techniques instead of HC1 and HC2. To our knowledge, no detailed evaluations regarding the footprint and performance of these compression mechanisms have been published in the literature to date.

A.4.3 Leveraging upon 6LoWPAN to realize the IoT

As described in the previous sections, the core specifications of 6LoWPAN are RFC 4944, RFC 6282 and RFC 6775. As 6LoWPAN is part of a full protocol stack, it is used in combination with different protocols, from the physical to the application layer. Further standardization work and research efforts are therefore focusing on, amongst others, improvements to the core specifications, extensions for non-IEEE 802.15.4 networks and the adoption of 6LoWPAN in practical use cases.

A.4.3.1 Improvements to core specifications

In addition to the core specifications, other sub-protocols are being discussed in the working group. As was already mentioned, the improved compression mechanism specified in RFC 6282 introduces a next header identifier to enable any arbitrary

next header compression. The *Generic Compression of Headers and Header-like payloads* (draft-bormann-6lowpan-ghc-05) [24] is one of the internet drafts that utilizes the identifier to perform compression of arbitrary headers.

Adaptation Layer Fragmentation Indication is another area that is under discussion. The draft, (draft-bormann-intarea-alfi), proposes a mechanism to indicate the presence of fragmentation at the adaptation layer and to indicate preferred MTU for the communication. This draft aims at improving application performance by limiting packet sizes to the smallest possible size to avoid fragmentation on link layers with small MTU values.

A.4.3.2 6LoWPAN over Non IEEE 802.15.4 technologies

The working group also adopted internet draft-ietf-6lowpan-btle, which applies 6LoWPAN technology to Bluetooth Low Energy. This draft is expected to be a companion of RFC 4944 by removing unnecessary features such as Mesh header and Fragmentation. This extension is the first draft that extends 6LoWPAN to non-IEEE 802.15.4 networks. Draft-mariager-6lowpan-v6over-dect-ule, is another draft proposed to extend 6LoWPAN to another non-IEEE 802.15.4 technologies. The draft proposes 6LoWPAN technology for DECT ULE (*Digital Enhanced Cordless Technology Ultra Low Energy*). The last two drafts indicate that the working group is considering link-layer technologies other than 802.15.4 to use with 6LoWPAN. As such, it is explored how 6LoWPAN can operate over heterogeneous low-power technologies, in a similar way as how IP can operate over different underlying technologies.

A.4.3.3 Adoption of 6LoWPAN in real life use cases

As 6LoWPAN is a key component in order to realize the IP-based integration of constrained devices, it is used in a multitude of projects, exploring a wide range of use cases such as smart infrastructures, smart buildings, smart environments, smart cities, ... In all cases, constrained devices, forming 6LoWPAN networks, are used to collect information from the real world and this information is used to generate intelligence and make the world around us smarter. Enumerating them all would be impossible, but in order to illustrate some application domains a limited number of specific examples is listed.

The Smart Energy and Home Automation: Restful Architecture (Sahara) project aims at using web-services for smart energy and home automation. The project uses 6LoWPAN on IEEE802.15.4, CoAP and other IETF standardized protocols [25]. Similarly, the European Union FP7 HOBNET project deploys an IPv6/6LoWPAN infrastructure to be used for the automation and energy management of smart and green building [26]. Another example is the FP7 Outsmart project [27], where 6LoWPAN networks are used for instance to optimize waste management [28].

CALIPSO is another FP7 project that aims at building IP-connected networks of smart object in the area of smart infrastructure, smart cities and smart toys by utilizing IETF standards (including 6LoWPAN) and other start-ups [29]. Finally, in [30] 6LoWPAN networks are being used for livestock monitoring applications and other similar use cases.

A.4.3.4 Other efforts

Management of (6LoWPAN) networks is another important aspect of 6LoWPAN that is being investigated. In line with this, 6LoWPAN MIB (draft-schoenw-6lowpan-mib) [31] is being proposed. The draft demonstrates a JSON format using a version of the Management Information Base database for management purposes along with the normal SMIv2 format.

From the previous discussion, it has become clear that addressing is also a key aspect when extending IPv6 to the constrained world through the use of 6LoWPAN. In this area, other IETF working groups are also conducting research activities in the area. The IETF Working group AUTOCONF (*Ad hoc* Network Autoconfiguration) is also working on Neighbor Discovery and stateless address autoconfiguration. Finally, as 6LoWPAN is meant as an enabler to bring IPv6 functionality to constrained networks, many higher layer protocols, such as routing, will run on top of 6LoWPAN networks. During the design of these solutions, the relation and interaction with 6LoWPAN is considered. For example, the routing protocols and mechanisms being developed by ROLL working group can also be seen as other 6LoWPAN related efforts [20]. The ROLL working group is discussed in the next section.

A.4.4 Research challenges

One of the key issues in connecting constrained networks with the Internet is fragmentation, caused by the 127 byte MTU of IEEE 802.15.4 versus the minimum MTU of 1280 bytes of links in IPv6 networks. Avoiding low-level fragmentation is important and requires knowledge about the MTU (*i.e.*, via discovery), knowledge about other protocols (type of routing) and interaction with the applications (and the potential use of fragmentation at this layer, such as block transfers in CoAP as will be discussed later). A good understanding and evaluation of the impact of fragmentation and solutions to avoid it is needed. For example, when using route-over solutions, additional buffer space is needed for packet reassembly and fragmentation at each intermediate node, since the information needed for routing can only be found in the first fragment [32]. The need of this additional buffer space can be avoided, by providing a small buffer to store the first few bytes of each packet. As the delivery of the first fragment could be handled by looking at the destination address, subsequent fragments may be routed based on the datagram-

tag which is stored in the small buffer we set aside. Different mechanisms may be considered for first fragments of a packet arriving late.

Regarding compression, advanced compression techniques have been proposed, including the compression of next headers. Until now, a thorough evaluation of the performance of these mechanisms for various communication patterns, address sizes and setups is missing. Transport layer compression methods are currently defined for UDP only. The 6LoWPAN working group is discussing generic next header protocol compression. Nevertheless, no compression mechanism is defined for TCP and ICMPv6 yet. Such a mechanism for TCP could be considered irrelevant for the majority of constrained devices, since they typically use UDP with an application-layer protocol such as CoAP that provides some of the missing features of TCP. However the use of CoAP on top of TCP has been explored [33].

There are still a number of other issues to be investigated. Layer 3 route-over and layer 2 mesh-under routing protocols are supported on 6LoWPAN networks. Mixing these protocols in a given 6LoWPAN network might be a path worth exploring [21].

Finally, the 6LoWPAN working group has already published a proposed standard document relating to Neighbor Discovery. Yet, Secure Neighbor Discovery is still unexplored territory. IPv6-like security up to level of constrained devices is being researched. Despite several efforts, key distribution to constrained devices remains one of the biggest challenges so far.

A.5 IETF ROLL Working Group

A.5.1 Group description and key protocols

A.5.1.1 Description

The specific properties of LLNs imply specific routing requirements for these networks. The ROLL working group [34] focuses on building routing solutions for LLNs because evaluation of existing routing protocols like OSPF, IS-IS, AODV, and OLSR indicate that they do not satisfy all of the specific routing requirements. More specific, the working group focuses on industrial (RFC 5673) [35], connected home (RFC 5826) [36], building (RFC 5867) [37] and urban sensor networks (RFC 5548) [38] for which different routing requirements were specified.

The working group focuses only on an IPv6 routing architectural framework while also taking into account high reliability in presence of time varying loss characteristics and connectivity with low-power operated devices with limited memory and CPU in large scale networks. The main realization of this working group is the design of the IPv6 route-over Routing Protocol for LLNs, also called RPL, which covers the routing requirements of all the application domains.

A.5.1.2 IPv6 Routing Protocol for Low Power and Lossy Networks

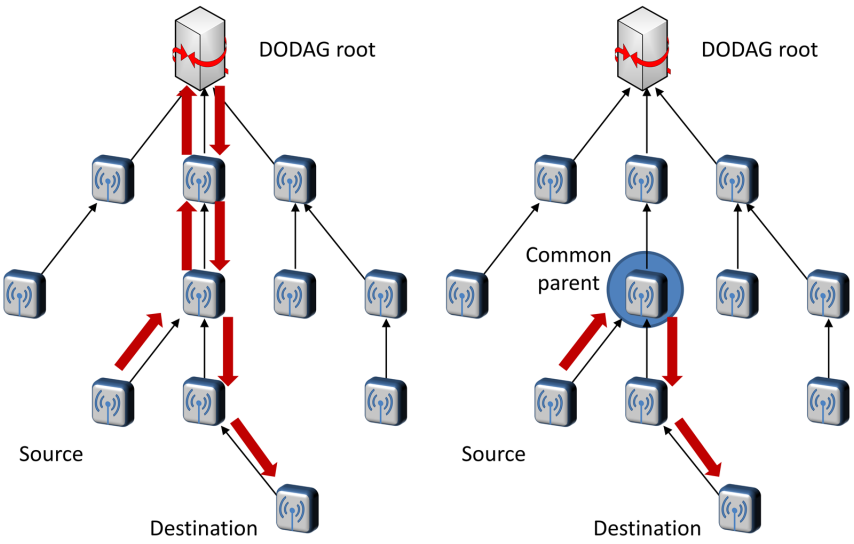
With the specification of RPL in *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks* (RFC 6550) [39], the IETF has specified a proactive "route-over" architecture where routing and forwarding is implemented at the network layer, according to the IP architecture. The protocol provides a mechanism whereby multipoint-to-point, point-to-multipoint and point-to-point traffic are supported. Although RPL was specified according to the routing requirements for LLNs, its use is not limited to these applications. RPL routes are optimized for traffic to or from a root that acts as a sink/root for the topology.

The functioning of the RPL routing protocol is based on the construction of a *Directed Acyclic Graph* (DAG), which consist of one or more DODAGs (*Destination Oriented DAGs*), for each sink/root a DODAG. The position of an individual node in a DODAG is determined by the rank of the node. This rank is calculated based on the *Objective Function* (OF), which defines how to translate one or more metrics and constraints, defined in *Routing metrics used for path calculation in low power and lossy networks* (RFC 6551) [40], into a rank. The OF also specifies how a node has to select his parent. Different DODAGs based on a same OF are represented by a RPLInstanceID. More details concerning OFs can be found in *Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)* (RFC 6552) [41], *The Minimum Rank with Hysteresis Objective Function* (RFC 6719) [42] and RFC 6551.

When a new node joins a RPL network, it first listens to receive *DODAG Information Object* (DIO) messages, which are broadcasted periodically when the trickle timer [43] of neighboring nodes fires. When no DIO message is received, the node will broadcast a *DODAG Information Solicitation* (DIS) message, which will force the neighboring nodes to send a DIO message. Based on the information of the DIO message from the neighboring nodes, the *Objective Function* (OF) will select the preferred parent. When every node in the network has selected a preferred parent, the DODAG (for a specific OF and for each sink) has been constructed. Routing from a node towards the sink is established by forwarding each message, for collection by the sink, to each nodes parent, until packets reach the sink.

Downward routes (root to leaf) are constructed using *Destination Advertisement Object* (DAO) messages. When a node has selected a preferred parent it will send a DAO message to his preferred parent, which will be forwarded, via the parent's parent, towards the root. Two modes of operation are supported: Storing (fully stateful) or Non-Storing (fully source routed) mode.

The mode of operation, for construction of downward routes will also influence the operation for point-to-point routes (Figure A.9). In the Non-Storing case, the packet will travel all the way to a DODAG root before traveling down. In the Storing case, the packet may be directed down towards the destination by a com-



(a) P2P traffic in non-storing mode.

(b) P2P traffic in storing mode.

Figure A.9: Packet flow for point-to-point traffic between two nodes in an RPL network.

mon ancestor of the source and the destination prior to reaching a DODAG root. If the destination is on the route towards the root, the destination node of course will not forward the message.

RPL messages are encapsulated into a new ICMPv6 message, defined in *Internet Control Message Protocol (ICMPv6) for IPv6 Specification* (RFC 4443) [44]. For RPL control messages this results in a message structure illustrated by Figure A.10, consisting of an ICMPv6 header followed by the actual message body (base) and some (optional) options.

1 byte	1 byte	2 bytes
Type = RPL	Code	Checksum

Figure A.10: ICMPv6 header for RPL control messages.

The code field identifies the type of the RPL control messages. Table A.4 gives an overview of the different codes and their corresponding RPL message type.

When the high order bit (0x80) in the code field is set, it indicates that security is enabled. In this case between the header and the base field a security field will be added.

In a DODAG Information Solicitation message the base field will consist of a reserved 1 byte flag field, a 1 byte reserved field. Together, with the flag and

Table A.4: Code fields in RPL ICMPv6 messages.

Code Field	RPL Message Type
0x00	DODAG Information Solicitation (DIS)
0x01	DODAG Information Object (DIO)
0x02	Destination Advertisement Object (DAO)
0x03	Destination Advertisement Object Acknowledgment
0x80	Secure DODAG Information Solicitation
0x81	Secure DODAG Information Object (DIO)
0x82	Secure Destination Advertisement Object (DAO)
0x83	Secure Destination Advertisement Object Acknowledgment

reserved field, unassigned bits must be set to zero on transmission and must be ignored on reception.

The DODAG Information Object (Figure A.11) contains information that allows receiving nodes to learn and configure for joining a DODAG. The message can indicate if the DODAG is grounded (1 bit G-field), which *mode of operation* (MOP) is followed and how preferable the root of this DODAG is compared to other DODAG roots (Prf-field). The message also contains the version number, the RPLInstanceID, DODAGID (128-bit IPv6 address that uniquely identifies a DODAG), a number of flags and the rank of the sending node.

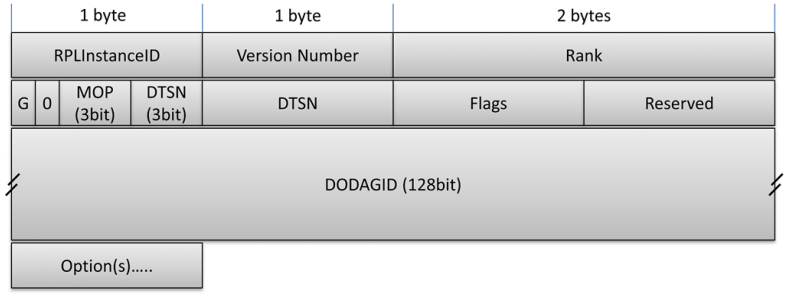


Figure A.11: The DODAG Information Object (DIO) Base Object.

The *Destination Advertisement Object* (DAO) message is illustrated in Figure A.12. The RPLInstanceID, learned from the DIO, is copied into the DAO message. When the sender expect the receiver to send a DAO-ACK the K-flag is set. To indicate the presence of the DODAGID field, the D-flag is used. The DAOSequence indicates the incremented sequence number incremented every time a node sends out a unique DAO message.

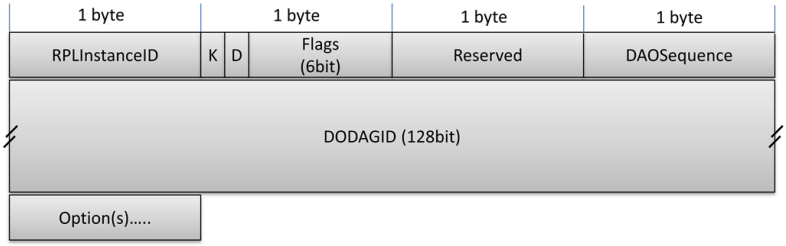


Figure A.12: The DAO Base Object.

A.5.2 Implementation and evaluation

A.5.2.1 Implementation

Already during the standardization process of the RPL protocol, the draft of the protocol was implemented in several platforms and simulators. An overview of the different implementations is presented in Table A.5.

Table A.5: Implementations incorporating RPL.

Name	OS	Protocol Version	Notes (Extensions, ..)
TinyRPL [45]	TinyOS	draft-ietf-roll-rpl-17	- uses BLIP 2.0 - only storing mode - only single RPLInstanceID - security options not supported - only telosb and epic platform support
ContikiRPL [46]	Contiki	RFC 6550	by default enabled on Tmote sky platform
OpenWSN [47]	OpenWSN	RFC 6550	
Nano-RK [48]	Nano-RK	draft-ietf-roll-rpl-07	
NanoQplus [49]	NanoQplus	draft-ietf-roll-rpl-13	

In Table A.6 an overview is given of the network simulators that implement the RPL protocol. In the table the TOSSIM simulator, standard for TinyOS, is not mentioned because TOSSIM requires micaz support, which is not available for TinyRPL. Several research papers such as [50] and [51] also implemented the RPL protocol into the WSNet simulator. In [51] this implementation is based on draft-ietf-roll-rpl-05.

Table A.6: Simulators incorporating RPL.

Name	Language	Protocol Version	Notes (Extensions, ..)
Cooja [52]	C with limited libs	RFC 6550	MSPsim (TinyOS + Contiki)
NS-3 [53]	C++ and Python	draft-ietf-roll-rpl-19	
OMNET++/ Castalia [54]	C++ (wrapped together with NED)	draft-ietf-roll-rpl-19	
J-SIM [55]	Tcl/Java	draft-ietf-roll-rpl-19	EU-funded FP7 ICT-257245 VITRO project

A.5.2.2 Using the protocol

RPL is designed to be widely applicable; therefore many configuration options are available. Based on experience, *Recommendations for Efficient Implementation of RPL* (draft-gnawali-roll-rpl-recommendations-04) [56] presents different design choices and configuration parameters that envision an efficient RPL implementation and operation. In *Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL)* (RFC 6687) [57], an overview and evaluation is given of how the protocol can handle two different use cases (a small outdoor deployment of sensor nodes for building automation and a large-scale smart meter network) to meet the desired requirements.

It is worth noting that a difference in design choices and configuration parameters can lead to interoperability problems. In [58] the interoperability between ContikiRPL and TinyRPL is investigated. The paper shows that, despite having two good performing independent implementations, a combination can lead to large scale differences in behavior when combining them in a mixed set up. It is also illustrated that subtle difference in lower layers can affect the system performance in unexpected ways.

A.5.2.3 Sensor-to-Sensor traffic

The routing for sensor-to-sensor communication is based on the communication paths between the root and the sensors. These paths are initiated by the communication of DAO-messages to their preferred parent. According to [59] this path setup can lead to congestion on the nodes around the root. In addition, the limitations on packet sizes for constrained devices for non-storing mode introduce the risk of fragmentation for paths with multiple hops. In contrast, for operation in storing mode, the limited memory influences the number of paths a node close to the root can store.

The authors of [60] state that the importance of point-to-point traffic flows in low-power and lossy networks is underestimated. In the paper it is demonstrated

for a network consisting of about 1,000 nodes that the shortest cost *peer-to-peer* (P2P) routes performs significantly better than the current RPL standard (using up and down P2P routes). This illustrates the need of additional point-to-point routing mechanisms. This conclusion is also confirmed by [61]. In this paper a solution is provided, called P2P-RPL, which extents RPL and performs better in a network of 27 fixed nodes running Contiki that has an average node degree of 4.39. While data packets in standard RPL traverse approximately 5 links on average, the links that are traversed when using P2P-RPL are halved for the same network. For even deeper nested DODAG trees, even higher gains are expected. P2P-RPL also decrease the traffic load around the sink: in storing mode with standard RPL 74,53% of the routes traverse the root, while for P2P-RPL this is only 16,03%.

A.5.2.4 Multipoint-to-Point traffic

In many IoT use cases, different sensors send their sensed data to a central sink. For this type of traffic RPL requires very limited control overhead. This overhead is further reduced by the use of the Trickle timer [43], which decreases the frequency of the sending of DIO messages when the network is stable. Also the responsibility of maintaining routes from leafs to the sink is delegated to each node by only selecting a parent which is closer (according to the OF) to the sink.

A.5.2.5 Multicast

Possibilities for the usage of RPL routing of multicast messages are stated in *Multicast Protocol for Low power and Lossy Networks (MPL)* (draft-ietf-roll-trickle-mcast-02) [62]. According to [63] the proposed draft solution has the advantage that it will work without modifications and reliability is increased by the per datagram state information maintenance, but it also has a number of drawbacks. A first drawback is that, instead of storing only destination information, for each packet a state has to be stored, which can result in scalability issues. The use of caching of messages, to avoid duplicates, can possibly improve the performance. Other drawbacks are an increase in complexity, susceptibility for out-of-order datagram arrival and energy and bandwidth inefficiencies due to the forwarding of messages to all parts of the network instead of only to parts with interested nodes (lack of group registration). To solve these issues, [63] introduces an alternative protocol, called SMRF.

A.5.2.6 Anycast

Currently, no support for anycast is provided in RPL. The use of anycast could be very efficient, especially when multiple sinks are available.

A.5.2.7 Link estimation

The estimation of the link quality with neighboring nodes is done by datapath validation via *Expected Transmission Count* (ETX) link cost estimation. The ETX equals the average number of transmissions for a packet to be successfully delivered to its next hop. The current selection mechanism prefers parents with the lowest rank. When there is more than one parent with the lowest rank, the first node is chosen as preferred parent, this has the advantage that no energy is consumed for evaluating the link quality to other neighboring nodes. As a downside, it only evaluates the currently used link; no alternative paths via newly discovered links are investigated. After some time this can lead to a sub-optimal routing topology. In [64] the solution of passive probing is introduced. Hereby the quality of a newly discovered link is initialized with the best value. This will force nodes to investigate all neighboring nodes as possible parent. At startup this solution will require more energy, but this energy is used to investigate the most optimal path, which can lead to a better overall energy efficiency.

However, in dense networks, the limited node memory results in constant re-evaluation of neighboring nodes, because neighbors in the neighbor table are continuously replaced by more recently used or heard neighbors. The authors of [64] suggest to use cache management to solve rediscovery and re-evaluation.

End-to-end link quality for a point-to-point route is currently not monitored in the RPL framework. A point-to-point link and its quality are currently determined by the individual links between the intermediate nodes. The nodes decide when to switch to a different parent based on the objective function. In *A Mechanism to Measure the Routing Metrics along a Point-to-point Route in a Low Power and Lossy Network* (draft-ietf-roll-p2p-measurement-07) [65] a mechanism is described to analyze the quality of the current route and to allow the router to initiate the discovery of a better route.

Similar to the previous draft, an on demand discovery mechanism for routes with specified metric constraints is presented in *Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks* (draft-ietf-roll-p2p-rpl-15) [66].

A.5.2.8 General performance

For the implementation of RPL in Contiki, according to [46], 3224 bytes of the ROM are used and 800 bytes of the RAM. For a Tmote Sky mote this implies 6.5% of the ROM- and 8% of the RAM-resources.

The usage of the Trickle timer helps to minimize the amount of routing overhead. In a stable network the Trickle timer allows the beacon intervals to exponentially increase and when noticeable changes in the network conditions are detected the beacon interval quickly decreases to the minimum interval. Still, for scenarios with bidirectional traffic, the RPL protocol generates a larger traffic over-

head (albeit with comparable delivery ratios) when compared to a protocol like LOAD [67]. It also has to be noted that LOAD is a reactive protocol, which has the additional advantage that the overhead is dependent of the traffic load, while RPL is proactive and has a constant overhead. When comparing TinyRPL with CTP [68], the well-known point-to-sink routing protocol for TinyOS, simulations indicate that the performance of both protocols has a comparable packet reception ratio and overhead. The benefit of RPL, compared to CTP, is the support for various types of traffic patterns (*i.e.*, multipoint-to-point, point-to-multipoint and point-to-point traffic) and the ability to directly connect to Internet nodes.

Finally, [69] analyzes the performance of RPL based on different simulations and concludes that RPL can ensure a fast network setup, which makes it a candidate protocol for monitoring applications in critical conditions. However they conclude that optimizations are required concerning the signaling in order to decrease protocol overhead.

A.5.3 Leveraging upon RPL to realize the IoT

A.5.3.1 Real life use cases

The RPL framework is currently used in different research projects. However in most cases the protocol is adapted to the specific requirements of the network. Most often, the RPL framework is used as a basis for the development of a specific protocol. Examples of such adaptations can be found in the previous evaluation section.

Additional examples include the following: in [70] the implementation of a smart monitoring system over a wireless sensor network is presented. The implementation focuses on the use of RPL to create an efficient and reliable routing structure. The paper also shows how, by changing some key parameters, the performance of RPL routing in a smart grid scenario can be influenced. In [71] a case study for the usage of RPL in an agricultural context is presented. Further, in [72] an example of the usage of RPL in transport logistics can be found.

A.5.3.2 Loop-free Repair Mechanisms

When link quality decreases and/or failure of a parent occurs, the repair mechanism of RPL, can introduce DODAG loops. In *Loop Free DODAG Local Repair* (draft-guo-roll-loop-free-dodag-repair-00) [73] an adaptation to the repair mechanism is proposed. In *Loop Free RPL* (draft-guo-roll-loop-free-rpl-01) [74] adaptation to the rank mechanism and objective function are presented. These drafts envision a loop free repair mechanism.

A.5.3.3 Heterogeneity

RPL specifications recommend forcing nodes with more constrained characteristics to operate only as a leaf node in a RPL network. A solution to force storing mode nodes to act as a non-storing mode in the presence of non-storing modes is presented in *RPL Routing Pathology In a Network With a Mix of Nodes Operating in Storing and Non-Storing Modes* (draft-ko-roll-mix-network-pathology-01) [75].

Most often, a simple objective function for a RPL based network is used. Objective functions however can also include device information to enforce specific routes. In *The Node Ability of Participation (NAP)* (draft-baryun-roll-nap-00) [76] the incorporation of node ability information is proposed to enable heterogeneity in terms of device capabilities.

A.5.3.4 DIS handling

In *DIS Modifications* (draft-goyal-roll-dis-modifications-01) [77] DIS options are presented to influence the control of responses to the solicitation for DIOs. A first option is the adding of a metric container field, which contains routing constraints a router must satisfy in order to respond, in a DIS message. A second option is the response spreading for preventing collisions with responses on a DIS multicast message. In this draft two cases (leaf node joining a DAG and identification of defunct DAGs) which make use of the DIS mechanism are described and compared.

A.5.4 Research challenges

Despite the standardization of RPL, some questions and gaps remain unsolved.

A.5.4.1 Interaction with MAC protocols

First of all, the RPL interaction with different MAC protocols and duty cycles is not yet determined. Especially the behavior at startup, when lots of configuration traffic is sent out, hasn't been researched yet. Since most IoT devices are, by definition, energy constrained, this aspect requires significant additional research.

A.5.4.2 Asymmetric links

The current RPL standard assumes the use of symmetrical links. For the selection of an optimal parent, the current RPL standard uses an approach based on the receipt of DIO messages from neighboring nodes. The node will select a parent based on the information in this message. However a node will use this link in the opposite direction of the receival of the DIO message. In case of asymmetric links or unidirectional links the node will not be able to reach his preferred parent,

which results in the reselection of another parent after some time. As such, this mechanism can be extended to also cope with asymmetric links.

A.5.4.3 Mobility

Currently RPL does not support node mobility. By supporting mobile nodes, the protocol can also be used for other application domains such as monitoring and safety systems for diverse traffic situations. In the field of mobile nodes in an RPL network, research activities are just starting. This means that lots of problems still have to be solved.

In [78] a multipath routing protocol for mobile sensor networks is presented. The protocol, called DMR, is based on the RPL framework. The performance of the protocol was compared against AODV and AOMDV by simulations in the NS-2.34 simulator. An improvement of energy efficiency with 25% and 20%, compared to AODV and AOMDV is achieved, while maintaining a delivery ratio of more than 97%.

Because a vehicular environment incorporates design elements of RPL, [79] provides a simulation performance study of adaptations of RPL for VANETs, by tuning different parameters of RPL. The test setup consists of nodes traversing in a line with an interdistance equal to the transmission range. The sink is positioned in the middle of the line. Information is transferred by multi-hopping with neighboring moving nodes.

In [80] adaptations are presented to enable data collection of a mobile node traversing, in a random pattern, in an environment of fixed nodes, running standard RPL. This is achieved mainly by having the mobile node continuously monitor his parents and neighborhood.

The selection and evaluation of the preferred parent, for mobile nodes, is still an open issue. For mobile nodes the link quality and neighboring nodes will vary due to the movement of the node. Therefore constant discovery of new neighbors and constant link analysis will be necessary. Because estimations are based on measurements, they also include an estimation of the past link quality. For fixed networks this estimation will be strongly correlated with the current link quality. For mobile nodes this estimation is typically outdated. An adaptive configuration for link estimation could help solve this problem, for instance by taking into account the movement (direction, speed,...) of the node, by reasoning on the succession of estimations (increasing values can indicate a node is approaching a parent or decrease of values a the opposite) or by adapting monitoring and/or discovery of neighbors.

Choosing a parent is important for reliably sending and receiving information. For fixed networks the parent with the best (stable) link quality is indeed the best choice. For mobile nodes, switching too frequently to a different parent also influences the reliability and energy consumption (extra control traffic for downward

traffic). Therefore a selection of a parent which will be a good parent for a longer time can be a better choice.

A.5.4.4 Multi-Sink support

In the standard, the possibilities for multi-sinks are briefly mentioned. However, currently no complete implementations are publicly available to our knowledge.

A.5.4.5 Scalability of the Non-Storing RPL approach

As already mentioned the non-storing mode is a good mechanism to limit the usage of memory for nodes in dense networks, but it also has a negative effect on the depth of routing tree. In non-storing mode, the addresses of all the intermediate nodes are added to the packet. This includes the danger that these messages will get fragmented and limits the available payload [60]. This is definitely a challenge that has not been investigated thoroughly.

A.6 IETF CoRE Working Group

More recent, in 2010, an IETF working group, called Constrained RESTful Environments (CoRE), was founded specifically to work on the standardization of a framework for resource-oriented applications, allowing realization of RESTful embedded web services in a similar way as traditional web services, but suitable for the most constrained nodes and networks. Their work resulted in the Constrained Application Protocol (CoAP), a specialized RESTful web transfer protocol for use with constrained networks and nodes.

A.6.1 Key protocols

CoAP is defined in draft-ietf-core-coap [81] in conjunction with a number of additional specifications. At the time of writing this article, there were three CoRE Internet-Drafts that are currently nearing completion and one completed RFC. In addition there were one more group Draft and 35 other individual Internet-Drafts listed on the CoRE website [5]. In this subsection we will describe briefly the (almost) completed CoRE Internet-Drafts and the published RFC. In the next section we will provide a brief summary of the topics covered by the individual Internet-Drafts in the CoRE working group.

A.6.1.1 Base CoAP

CoAP uses the same RESTful principles as HTTP, but it is much lighter so that it can be run on constrained devices [82, 83]. To achieve this, CoAP has a much

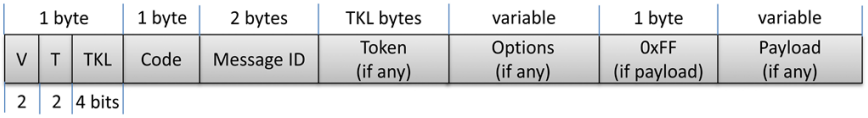


Figure A.13: CoAP Message Format consisting of a 4-bytes base binary header followed by optional extensions

lower header overhead and parsing complexity than HTTP. It uses a 4-bytes base binary header that may be followed by compact binary options and payload. Figure A.13 shows the CoAP message format as specified in version 13 of the draft. This version introduced a breaking change in the message format. However, it is expected that this will be the final change of the format.

The CoAP interaction model is similar to the client/server model of HTTP. A client can send a CoAP request, requesting an action specified by a method code (GET, PUT, POST or DELETE) on a resource (identified by a URI) on a server. The CoAP server processes the request and sends back a response containing a response code and payload. Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport such as UDP and thus it also supports multicast CoAP requests. This allows CoAP to be used for point-to-multipoint interactions which are commonly required in automation. Optional reliability is supported within CoAP itself by using a simple stop-and-wait reliability mechanism upon request. Secure communication is also supported through the optional use of Datagram Transport Layer Security (DTLS).

As can be seen in Figure A.13 all CoAP messages start with a 4-bytes base binary header that consists of the following fields:

- **Version (V)**: A 2-bit unsigned integer indicating the CoAP version number. Current version is 1. Other values are reserved for future versions.

- **Type (T)**: A 2-bit unsigned integer indicating if this message is of type Confirmable (0), Non-Confirmable (1), Acknowledgement (2) or Reset (3).

- **Token Length (TKL)**: A 4-bit unsigned integer indicating the length of the variable-length Token field (0-8 bytes). Lengths 9-15 are reserved.

- **Code**: An 8-bit unsigned integer indicating if the message carries a request (1–31) or a response (64–191), or is empty (0). (All other code values are reserved.) In case of a request, the Code field indicates the Request Method (1: GET; 2: POST; 3: PUT; 4: DELETE); in case of a response a Response Code. Possible values are maintained in the CoAP Code Registry (see section 12 of the draft).

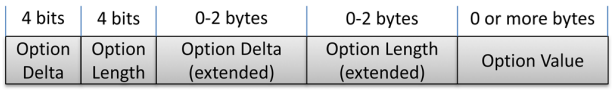


Figure A.14: CoAP Option Format

- **Message ID**: A 16-bit unsigned integer in network byte order used for the detection of message duplication, and to match messages of type Acknowledgement/Reset to messages of type Confirmable/ Non-confirmable.

The base 4-bytes header may be followed by one or more of the following optional fields:

- **Token**: 0 to 8 bytes, as given by the Token Length field. The Token value is used to correlate requests and responses. The rules for generating a Token and correlating requests and responses are defined in Section 5 of the draft.
- **Options**: An Option can be followed by the end of the message, by another Option, or by the Payload Marker and the payload. The format of the Options field is shown in Figure A.14 and is described in more detail in the next paragraph.
- **Payload**: If present and of non-zero length, it is prefixed by a fixed, one-byte Payload Marker (0xFF) which indicates the end of options and the start of the payload. The payload data extends from after the marker to the end of the UDP datagram, *i.e.*, the Payload Length is calculated from the datagram size. The absence of the Payload Marker denotes a zero-length payload.

To be able to offer communication needs that cannot be satisfied by the base binary header alone, CoAP defines a number of options which can be included in a message. Each option instance in a message specifies the Option Number of the defined CoAP option. Instead of specifying the Option Number directly, the instances must appear in order of their Option Numbers and a delta encoding is used between them (The Option Number for each instance is calculated as the sum of its delta and the Option Number of the preceding instance in the message. For the first instance in a message, a preceding option instance with Option Number zero is assumed. Multiple instances of the same option can be included by using a delta of zero). The fields in an option are:

- **Option Delta**: 4-bit unsigned integer. A value between 0 and 12 indicates the Option Delta. A value of 13 indicates that an 8-bit unsigned integer follows the initial byte and indicates the Option Delta minus 13. A value of 14 indicates that a 16-bit unsigned integer in network byte order follows the initial byte and indicates the Option Delta minus 269. The value 15 is

reserved for the Payload Marker and cannot be used here. The resulting Option Delta is used as the difference between the Option Number of this option and that of the previous option (or zero for the first option).

- **Option Length**: 4-bit unsigned integer. A value between 0 and 12 indicates the length of the Option Value, in bytes. A value of 13 indicates that an 8-bit unsigned integer precedes the Option Value and indicates the Option Length minus 13. A value of 14 indicates that a 16-bit unsigned integer in network byte order precedes the Option Value and indicates the Option Length minus 269. The value 15 is reserved for future use.

- **Value**: A sequence of exactly Option Length bytes. The length and format of the Option Value depend on the respective option, which may define variable length values.

As an example of a simple CoAP option consider the Content-Format option. This option indicates the representation format of the message payload. This option has the Option Number 12 and its Option Length is between zero and two bytes. The Option Value itself is an unsigned integer that is defined in the CoAP Content Format registry (Section 12 of the draft).

The IETF CoRE working group considers the constrained restful environments as an extension of the current web architecture. The group envisions that CoAP will complement HTTP and that CoAP will be used not only between constrained devices and between servers and devices in the constrained environment, but also between servers and devices across the Internet [84]. An important requirement of the CoRE working group is to ensure a simple mapping between HTTP and CoAP so that the protocols can be proxied transparently. Thus proxies and/or gateways play a central role in the constrained environments architecture. These proxies have to be able to communicate between the Internet protocol stack and the constrained environments protocol stack and to translate between them as needed.

A.6.1.2 CoRE Link Format

Resource discovery is important for machine-to-machine (M2M) interactions, and is supported in CoAP using the *CoRE Link Format* as described in RFC 6690 [85]. A well-known relative URI "/.well-known/core" is defined as a default entry-point for requesting the list of links about resources hosted by a server. Once the list of available resources is obtained from the server, the client can send further requests to obtain the value of a certain resource. The example in Figure A.15 shows a client requesting the list of the available resources on the server (GET/.well-known/core). The returned list shows that the server has, amongst others, a resource called/s/t that would return back the temperature in degrees Celsius. The client then requests the value of this resource (GET/s/t) and gets a reply back from the server (23.5C).

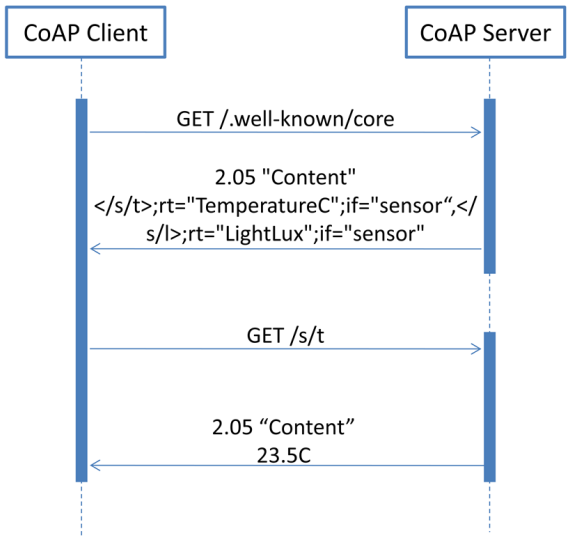


Figure A.15: An example of Constrained RESTful Environments (CoRE) resource discovery and Constrained Application Protocol (CoAP) request

A.6.1.3 Block transfer

In many cases the payloads that CoAP needs to carry is very small (e.g., just a few bytes for temperature sensor, door lock status or toggling a light switch). In these cases the basic CoAP message provides very efficient means of communication and work very well. However in some cases CoAP needs to handle larger payloads (e.g., firmware update). Since CoAP is based on datagram transports such as UDP or DTLS, data fragmentation and reassembly is not offered by these transport protocols. Relying on IP fragmentation is also not very helpful, because fragmentation and reassembly does not perform well in LLN due to memory requirements imposed by route-over routing as described in Section A.5. Additionally, IP fragmentation can handle only payloads up to 64 KiB. Thus, providing a mechanism at the application layer that is able of transferring large amounts of data in smaller pieces becomes a necessity. This will not just help avoid the 64KiB UDP datagram limit, but also will help avoid both IP fragmentation (MTU of 1280 for IPv6) and also adaptation layer fragmentation in LLNs (60–80 bytes for 6LoWPAN).

To overcome the payload size limitation, draft-ietf-core-block defines two CoAP options: Block1 and Block2 [86]. By using this pair of options CoAP becomes capable of transferring a large payload in multiple smaller CoAP messages. Both Block1 and Block2 options can be present both in request and response messages. In either case, the Block1 Option pertains to the request payload, and the Block2 Option pertains to the response payload. An important aspect of this mechanism

is that often the server can handle block transfers in a stateless fashion—it does not require connection setup and the server does not need to track each transfer separately and thus conserve memory.

A.6.1.4 Observation of resource

The state of a CoAP resource can change over time. In draft-ietf-core-observe [87] a simple CoAP extension is defined that enables a server to inform interested clients about the state change. A client interested in observing a resource includes the observe option in its GET request to the server. Whenever there is a change of the resource state, the server sends a notification to the client. Also, in case the state of the resource does not change, but the time since the last notification exceeds the max-age value of the resource, a notification is sent. As such, observe offers the possibility for a client to have an up-to-date representation of the resource without the client having to constantly poll for changes. New resource states are transmitted from the server to the clients according to a best-effort approach. The observe protocol foresees mechanisms to ensure consistency between the state observed by each client and the actual resource state.

A.6.2 Implementation and evaluation

A.6.2.1 CoAP implementations

At the time of writing this article, the CoAP protocol still is not yet finalized. However it is considered in its final stages before being finalized. Nevertheless several implementations of the CoAP protocol for various platforms and programing languages already exist. Some of these implementations are open source and others have commercial licenses. Interoperability between many of these implementations has been formally tested by the *European Telecommunications Standards Institute* (ETSI), a non-profit standards organization. ETSI organizes a series of events called ETSI Plugtests to test interoperability of telecommunication technologies in a multi-vendor, multi-network or multi-service environment. ETSI Plugtests, the IPSO Alliance and the FP7 Probe-it project have organized two IoT CoAP Plugtests. In addition to assessing the interoperability of participating products, these Plugtests events aimed to validate the CORE base standards. The first Plugtests event (March 2012) was attended by 18 companies testing the world's first CoAP client and server implementations. The features tested at this event included the base CoAP specification, CoAP Block Transfer, CoAP Observation and the CoRE Link Format. More than 90% of 3,000 tests executed in this event were successful. According to ETSI Plugtests this result is classified as a high level of interoperability [88].

At the second IoT CoAP Plugtests event (November 2012), some additional tests were added to cover previously untested aspects of CoAP in addition to in-

troducing new optional tests for proxy functionality and M2M communication. In this event 1,775 test cases were performed, in 60 pairing test sessions, with a success rate of 97.8% [89]. The improvement in interoperability compared to the first IoT CoAP Plugtests event indicates that the base CoAP protocol and its main options are getting more robust.

With insights of the first IoT Plugtests a survey on the current state of the art of lightweight REST implementations was presented in [90]. In Table A.7 we present an adapted version of the survey results. In this adapted version we concentrate on publicly available implementations and servers and commercial implementations that are publicly announced (See [90] for other implementations). This table uses device classes as defined in [32]. Class 1 devices have ˜10 KiB of RAM, and ˜100 KiB of ROM. Class 2 devices have ˜50 KiB of RAM, and ˜250 KiB of ROM.

The existence of such a wide range of implementations across a broad range of programing languages and most importantly platforms (constrained and not) demonstrates the feasibility of the protocol implementation.

A.6.2.2 CoAP performance evaluation

Going beyond mere compatibility tests and the ability to implement the protocol on constrained devices, a few studies have been made in order to evaluate the protocol's behavior and suitability for certain IoT use cases. In particular the comparison between CoAP and HTTP has been studied a few times. The results of such comparisons are considered preliminary since the CoAP protocol itself is not fully standardized yet, and because the studies used the protocol at different stages of its development. However these results still provide a good indication of what to expect when the protocol is fully standardized and the used implementations are further optimized.

For example, the work described in [105] provides an evaluation of CoAP compared to HTTP in terms of mote's energy consumption and response time in wireless sensor networks. The results for energy consumption, obtained by simulations for a fixed 10 second client request interval, show that while receiving and processing packets, the energy consumed when using CoAP is approximately half compared to the one consumed when using HTTP. While transmitting packets, the energy required by CoAP is 4 times lower than the energy required by HTTP. On the other hand, while being in idle mode, CoAP and HTTP result in similar values of energy consumption. When testing the response time on real sensor motes, the results show that CoAP/UDP introduces 9 to 10 fold lower response times than HTTP/TCP.

Similarly [106] evaluates the performance of HTTP/TCP and CoAP/UDP over a duty cycled radio layer. With a small modification to the duty cycling layer the authors achieved great improvement in performance at retained low power consumption. Furthermore they introduced an in-network caching mechanism that

Table A.7: CoAP implementations. Please note that these implementations are work in progress since CoAP itself is work in progress.

Name/ Company	**License**	**Language**	**Platform**	**Notes**
Consorzio Ferrara Ricerche	[91]	NesC/C	TinyOS	Own "SiGLoWPAN" IPv6/6LoWPAN stack for Class 1 devices
Californium [92]/ ETH Zurich	3-clause BSD	Java	JVM	Framework for unconstrained devices; provides client, server, and proxy stubs
Copper [93]/ ETH Zurich	3-clause BSD	JavaScript	Firefox	Management and testing tool as a browser extension; focus on user interaction
Erbium [94] /ETH Zurich	3-clause BSD	C	Contiki	For class 1 devices such as sensor nodes
CoAP++ [95]/iMinds		C++	Click Modular Router	Framework for unconstrained devices; provides client, server, proxy and gateway
Evcoap [96]/KoanLogic	2-clause BSD	C	Linux	General purpose protocol implementation
Patavina Technologies [91]	Commercial	C++	proprietary OS	Wired and wireless embedded devices and sensor nodes; working on a port to uC/OS by Micrium
NanoService Device Library [97]/Sensinode	Commercial	C		OS-independent C library for Class 1 and 2 devices. Also available a JAVA SDK for unconstrained devices
libcoap [98]/ Universität Bremen TZI	GPLv2, 2-clause BSD	C	POSIX and Contiki	General purpose library for Class 1 and 2 devices and up
CoapBlip [99]/ Universität Bremen TZI	BSD-style	C	TinyOS	TinyOS-port of "libcoap"; runs on Class 1 devices.
coap.me [100]/ Universität Bremen TZI		Ruby		http://coap.me provides an HTTP front-end to crawl CoAP servers, and a CoAP server for interoperability testing
jCoAP [101]/ Universität Rostock	Apache 2.0	Java	JVM	For unconstrained devices; also targets mobile and embedded platforms
Scuola Superiore Sant'Anna [102]		Erika API	Erika OS	A middleware for building an infrastructure of wireless sensor nodes.
CoAPy [103]/ People Power	BSD	Python		Last updated on July 2010
CoAP in wiselib [104]/ wisebed project	GNU Lesser GPL v3	c++		Wiselib algorithm classes can be compiled for several sensor platforms such as iSense or Contiki, or the simulator Shawn.

significantly improves the performance of software updates in incrementally deployed sensor networks.

In a third evaluation [107] the author finds that CoAP/UDP perform better than HTTP/TCP for the intelligent cargo container use case he evaluated. In particular the author reports a 6 times lower message size and a 4 times lower *Round Trip Time* (RTT). This is mainly due to CoAPs compressed header and the avoidance of the TCP handshaking mechanisms. A further study of the same use case [108] compares using CoAP/UDP to HTTP/UDP in addition to the typical HTTP/TCP. The study shows that generally UDP based protocols perform better for constrained networks due to using lower number of messages when retrieving resources. When comparing both protocols (CoAP and HTTP) when run over UDP, the study shows that CoAP performs better than HTTP. CoAP also has the added value of optional reliability since it has its own simple retransmission capability.

A.6.3 Leveraging upon CoAP to realize the IoT

The base CoAP protocol along with the three main complementary extensions (block, observe and Core Link Format) provide a basic set of protocols to solve a wide range of communication needs in constrained environment. However, since the protocol tries to be minimalistic and yet extendable at the same time, a couple of needs remain unaddressed in these base protocols. To try to address the unsolved issues, a few individual Internet-Drafts were proposed in the CoRE working group. These drafts are in various states of development, with various levels of CoRE review and interest. A thorough overview of all these individual Internet-Drafts is given in draft-bormann-core- roadmap [109]. In this subsection we highlight some of these individual Internet-Drafts and complement them with additional literature that tries to leverage upon CoAP to realize the IoT.

A.6.3.1 Discovery and naming

Resource Directory (RD) servers provide a way to collect resource descriptions from multiple servers into one central location. To facilitate setting up such a RD, draft-shelby-core-resource-directory identifies needed protocol elements [110].

In [95], CoAP is used to facilitate discovery and deployment of constrained devices. The proposed approach makes use of CoAP and combines it with DNS in order to enable the use of user-friendly fully qualified domain names (FQDN) for addressing sensor nodes. It also includes the translation of HTTP to CoAP, thus making the sensor resources globally discoverable and accessible from any Internet-connected client using either IPv6 addresses or DNS names both via HTTP or CoAP. This approach can be enhanced by publishing the discovered resource to one or more RDs.

Another example of how DNS can be used in a hierarchical fashion to enable easier access to resources is described in draft-ietf-core-groupcomm [111]. Here access to groups of resources can be provided via the use of Group FQDN that are uniquely mapped to a site-local or global multicast IP address via DNS resolution. For example the Group FQDN all.bldg6.example.com would refer to *all nodes in building 6* and the Group FQDN all.west.bldg6.example.com would refer to *all nodes in the west wing of building 6*.

A.6.3.2 Congestion control

The base CoAP draft only defines a very basic congestion control when using reliable message transmissions and does not provide any congestion control when using the non-reliable transmissions mode, that is likely to carry the majority of traffic. To overcome this shortcoming a few proposals try to provide more advanced congestion control schemes. These proposals can generally provide more optimized performance in exchange for more implementation complexity and/or a narrower field of application. For example, draft-bormann-core-cocoa [112] defines some more advanced CoRE congestion control mechanisms. The main idea here is to provide a way to better estimate the RTT than that implied by the default initial timeout of 2 to 3 seconds. Further suggestions for the enhancements to this estimation of the RTT are presented in [113].

Another mechanism for congestion control is proposed in [114] by adding an option that allows a server to indicate its desire for some pacing of the requests sent to it by one client; enabling a form of server load control.

A.6.3.3 Advanced interaction patterns

The base CoAP provides good support for the simple interaction patterns between clients and servers. However more advanced interaction patterns such as the communication between a group of devices requires extensions to the base protocol. In fact, it is anticipated that constrained devices will often naturally operate in groups (e.g., all window shutters on a given side of building may need to be lowered or raised as a group). Draft-ietf-core-groupcomm [111] discusses fundamentals and use cases for group communication patterns with CoAP. Building upon IPv6 multicast capabilities, the draft describes how CoAP should be used in both constrained and unconstrained networks and provides guidance for deployment in various network topologies. Although this draft has been adapted as a working group draft, it is still (at least in certain parts) in an explorative mode and will require additional investigation before conclusive results become available.

The CoAP observe option allows clients to register with servers to be notified whenever the state of a resource changes [87], much like the publish/subscribe paradigm in conventional web services. In [115] a new CoAP option “Condition”

was proposed to extend the observe option. This option can be used by a CoAP client to specify the conditions the client is interested in. Several condition types, *i.e.*, filtering options, have been identified based on realistic use cases. Using conditional observations, the CoAP server will send a notification response with the latest state change only when the criterion is met. Using this mechanism a client can for example indicate that it is only interested in temperature values above 25 C and not in all state changes. The feasibility of implementing this conditional observe on a constrained device is evaluated and proven in [116]. The correct operation for a simple scenario showed that the use of conditional observations can result in a reduced number of packets and power consumption compared to that which is normally observed in combination with client-side filtering.

A.6.3.4 Communication with Sleepy Nodes

Sleepy Nodes are network nodes that sleep most of the time in order to save energy and thus achieve longer battery life times. The base CoAP standard assumes that the communication layers below the application provide support functions for sleeping nodes. Adding better support for sleepy nodes at the application layer might be able to further reduce the power requirements of these nodes. This support is currently a very active subject of discussion in the CoRE working group; this is apparent from the relatively high number (at least seven) of individual Internet Drafts in the group that try to address this issue in one way or another.

The base CoAP provides minimal support for sleepy nodes by supporting caching in intermediaries. Resources from a sleepy node may be available from a caching proxy (if previously retrieved) even though the node is asleep. This support is enhanced by using the observe option and thus allowing sleepy nodes to update caching intermediaries according to their own schedule.

Most of these proposals try to achieve better support for sleepy nodes either by extending the functionality of the intermediaries or by extending the CoAP observe option or by a combination of both. For example, [117] proposes to store the actual resource representations in a special type of RD called the Mirror Server. Clients can then fetch the resource from the Mirror Server regardless of the state of the sleepy server. One the other hand, by using the conditional observe option as proposed in [118], the nodes may be allowed to sleep even longer. Similarly the approach of [119] is to introduce storing of sleep characteristics in the RD. Clients can then query the RD to learn the sleep status of the sleepy node before attempting communications. Both [118] and [119] include using/extending the observe option as part of their overall approaches.

A related patent [120], describes inserting sleep information into a header option or into a payload of an application layer message. Whereas the application layer message may be conveyed in HTTP or CoAP.

A.6.3.5 Security

Security is another hot topic on the CoRE working group with many drafts trying to tackle various security aspects of the Things and the Information they reveal about the physical world. It is anticipated that most real deployments of the IoT will require security services (e.g., confidentiality, authentication, authorization). However it is also argued that there is no single security architecture for the IoT [121]. A good description of the Thing Lifecycle is provided in [122] along with resulting architectural considerations.

The authors of [123] present a three-phase protocol to bootstrap constrained devices in a wireless sensor network based on IPv6 and CoAP. The protocol phases include service discovery, distribution of security credentials, and application-specific node configuration.

CoAP proposes to use DTLS to provide end-to-end security to protect the IoT. However DTLS is a heavyweight protocol and its headers are too long to fit in a single IEEE802.15.4 MTU. The works presented in [124–127] look specifically into the use of DTLS in constrained networks from different angles. As an example, while [126] shows how to build minimal implementations of TLS, the approach used in [127] relies on providing 6LoWPAN header compression mechanisms to reduce the size of the DTLS security headers. The authors report as an example that the number of additional security bits needed for the DTLS Record header that is added in every DTLS packet, can be reduced by 62%.

Another relevant security protocol is the Internet Key Exchange version 2 (IKEv2) which is a used for setting up IPsec security associations. IKEv2 includes several optional features, which are not needed in minimal implementations. The Minimal IKEv2 draft [128] shows how to build minimal implementations of the security protocols IKEv2 for constrained environments.

A.6.3.6 Intermediaries

The base CoAP draft defines basic mapping between CoAP and HTTP. However it is expected that Intermediaries will continue to play a big role in the IoT and that the basic mapping needs to be enhanced to support advance features. Some ideas about these enhancements are presented in [129]. Additional useful examples for more advanced forms of mapping and usages are described in [130].

A.6.3.7 CoAP in cellular networks

The *Short Message Service* (SMS) of mobile cellular networks is frequently used in M2M communications. The service offers small packet sizes and high delays just as other typical types of LLNs. Since the design of CoAP takes the limitations of LLNs into account, it is expected that CoAP can be nicely used with SMS. The

adaptation of CoAP to the SMS transport mechanisms and the combination with IP transported over cellular networks is described in [131].

A.6.3.8 Real life use cases of CoAP in the IoT

Some recent publications show that CoAP is being considered as a good candidate to solve current issues in real life application of the IoT. For example, [108] presents an IP based solution to integrate sensor networks used in a cargo container with existing logistic processes, highlighting the use of CoAP for the retrieval of sensor data during land or sea transportation.

Kovatsch *et al*. [132] propose an IoT architecture where the infrastructure is agnostic of applications and application development is fully decoupled from the embedded domain. This is achieved by creating a common application layer that fosters the development of novel applications. The application logic of devices is running on application servers, while thin servers embedded into devices export only their elementary functionality using CoAP resources.

GlobalPlatform is a cross industry not-for-profit association which publishes specifications facilitating the secure and interoperable deployment and management of multiple embedded applications on secure chip technology. According to a recent presentation [133], GlobalPlatform is considering the use of CoAP for the management of secure environments and other aspects covered by GlobalPlatform standards.

Castro *et al*. [134] present how the IoT is integrated for improving terrestrial logistics offering a comprehensive and flexible architecture, with high scalability, according to the specific needs for reaching an item-level continuous monitoring solution. CoAP is used here to provide tracking and monitoring services at any time during the transportation of goods. The solution makes use of observe and blockwise transfer options to optimize data transfers.

Optimizing energy policies requires monitoring, analyzing, and controlling of power consumption. Smart metering is an emerging topic for realizing such modern energy policies. In this field a large number of proprietary and open standards for communication (with low or no interoperability to each other) exist today. Therefore, it is very difficult to integrate multi-vendor solutions using one sustainable holistic approach. To this end the authors of [33] propose to use Web Service technology as an open widespread Internet standard for the creation of a heterogeneous network for smart metering devices. This work uses CoAP over TCP transport instead of using it over UDP, which is still not specified in the current CoAP drafts. Nevertheless, traffic overhead was reduced by over 90% by using CoAP instead of HTTP and EXI for XML binary encoding.

Using CoAP and *REsource LOcation And Discovery* (RELOAD), [135] proposes a new architecture for wide area sensor and actuator networking. RELOAD is a P2P signaling protocol for use on the Internet that is currently being stan-

dardized by the IETF. The architecture provides a decentralized peer-to-peer rendezvous service for CoAP nodes in WSNs and enables a P2P federation of geographically distributed WSNs. This is achieved by the use of proxy nodes that are part of the WSN but also connect to a RELOAD overlay network via cellular Internet access. The authors conclude that such architecture is most beneficial for large-scale networks having from moderate to high levels of interdevice communication.

Rahman *et al*. [136] present a smart object gateway architecture that allows for efficient service delivery between the Smart Object and an endpoint on the Internet such as an application server. A survey of some other examples of how CoAP is been used to realize the IoT can be found in [137].

A.6.4 Research challenges

Since CoAP is not fully standardized yet, many related aspects remain to be examined before definite conclusions can be drawn. Some of these aspects are currently being researched and have been documented in Section A.6.3. Some other aspects have been identified but no solutions were proposed yet. In this section we summarize the main aspects of the CoAP protocol where we think that extensive research is still needed.

Although a few suggestions for congestion control were already proposed (see Section A.6.3.2), we think that this area requires more research attention. The primary reason being that UDP is the main transport mechanism for CoAP. Unlike TCP, UDP does not provide any reliability mechanism or congestion control. CoAP has its own optional light weight reliability mechanism, but virtually no congestion control. Simple mechanisms for congestion control, that are optimized for LLNs and that take into account the limited amount of resources available on constrained devices are still lacking.

Another research aspect is related to the MTU size. The IPv6 minimum MTU is 1,280 bytes, whilst the maximum MTU for IEEE 802.15.4 networks is 127 bytes. The 6LoWPAN adaptation layer provides an efficient fragmentation method to allow the transmission of larger packets. However large packet sizes still sometimes impose a big problem to the receiving party. If the received message does not fit in the input buffer it could cause unpredictable effects on the receiving side, e.g., operating system restarting because of buffer overflow. For this reason, it can be useful to specify during the resource discovery sequence the maximum accepted length of the response message with the resource description [138]. If both parties support the block option, agreeing on the optimal block transfer size in a way that avoids fragmentation and assembly of the packet at the lower layers will possibly have a high positive impact on the overall performance. This is something that needs further research and experimentation.

Security of the IoT remains a challenge despite the fact that several proposals have already been made to cover certain aspects of security. Issues such as secure distribution of encryption keys in LLNs still need to be explored.

A basic mapping between HTTP and CoAP is well defined in the core CoAP draft. However since CoAP is by design highly extendable and new options are being regularly added, such extensions might impose new challenges to the HTTP/-CoAP mapping. The same consideration also applies in general to all intermediaries, that need to be updated if they wish to understand new options as well (some basic information for intermediaries is contained in an option's option number, which allows intermediaries to correctly handle unknown options). Caching intermediaries have an even bigger challenge to find out and implement optimal caching strategies.

Group communication in a LLN context still needs more attention as well. The use of multicast satisfies many group communication needs, but not all of them. Multicasts are transported unreliable and since LLNs are lossy by nature, many applications will opt not to use multicast if they want to make sure that their messages are indeed delivered. For example, if multicast is used to turn on all the lights in a room, it wouldn't be acceptable if one light stays off because it did not receive the multicast message. Alternatives to the use of multicasts might be needed for such use cases. Maybe status synchronization of resources can also be used in certain use cases to make sure that the message did indeed get through. The efficiency of such and other approaches need to be examined.

Finally, some challenges are related to the fact that resources on constrained devices often need to be accessed by other machines as well as by humans. These two types of "clients" have somewhat contradictory requirements. Humans for example prefer easy to remember (and a bit lengthy) resource names, while M2M communication is better served by providing concise names. Some more research is needed in order to satisfy the needs of both types of "clients" without adding many requirements on the constrained devices. The same considerations also apply to the use of extended semantics, e.g., a *Web Service Definition Language* (WSDL). WSDL provides a machine-readable description of how a web service can be used, what parameters it expects, and what data structures it returns. However WSDL tends to become relatively verbose and long for constrained nodes to handle, thus alternatives need to be researched.

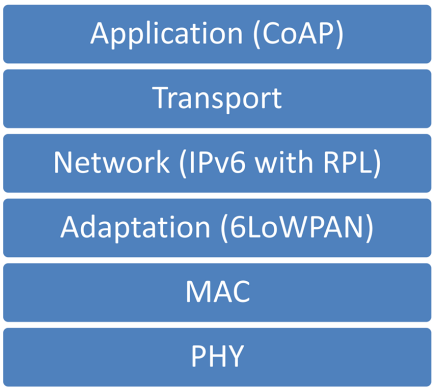


Figure A.16: IETF LLN protocol stack

A.7 Using IETF standards to realize the Internet of Things

A.7.1 Overview of the IETF LLN Protocol Stack

The previous sections provided an overview of different standards, their current status and open research topics. Combined, these individual standards form an *IETF LLN protocol stack* to support the realization of an interoperable Internet-of-things. In Figure A.16 a representation is given of how the different LLN standards fit together. Currently the entire IETF LLN protocol stack is available in the Contiki OS, which is used in, for example, the CALIPSO FP7 project [29]. For simulating the IETF LLN protocol stack OpenWSN [47] can also be used.

The LLN protocol stack provides end-to-end access to embedded web services, thus enabling new functionalities or building novel services involving IoT objects. This section focuses on the next steps: which additional (new or existing) research, protocols and/or standards are needed to realize a fully-automated, all-encompassing Internet of Things.

A.7.2 Realizing the Web of Things

To allow the integration of an increasingly large number of IoT devices, self-configuration protocols will be required. Solutions for self-configuration such as [95] allow newly deployed constrained devices to be automatically discovered, automatically assign DNS hostnames and transparently make the IoT resources directly accessible and browsable over IPv6 via HTTP or CoAP. Alternatively, devices can add their resources to a publicly accessible directory service. This sort of solution forms an important building block that facilitates the actual usage of em-

bedded web services (without human intervention) as is required for realizing the Web of Things (WoT). For instance, a client can automatically be notified about new resources and continuously observe the state of a resource using the CoAP observe extension [87], leading to a consistent representation of all resources of interest. Using conditional observations [115], interested parties can be notified about resource states that satisfy specific conditions, thereby acting as an enabler to build applications such as sensor—actuator interactions. These extensions enrich the capabilities of the basic CoAP protocol and contribute to the realization of the WoT.

Another important research domain focuses on web service composition whereby different IoT services are combined to realize complex goals. For instance, embedded web services can automatically be combined to create complex interaction scenarios where knowledge about the real world is used, linked with other services and processed to act again upon the physical world. Existing composition and orchestration frameworks such as described in [139], need to be extended in order to realize the WoT. The main challenge is the adaptation of existing web service composition models to take into account the limitations of constrained devices. Also, when time varying data from constrained objects is incorporated or web services act upon the real world, issues such as consistency, failures, correct execution of all transactions as described in [140] need to be explored in view of a constrained environment.

The link between the IoT and state-of-the-art cloud technology solutions is made clear in [141]. Cloud technology can be used for collecting, storing and processing the enormous amount of sensor data. Tiny objects can also be introduced as part of grid computing e.g., for the collection and processing of environmental information. In [142] an extensive overview of the introduction of mobile devices into Grid systems is given and an extension to the constrained world seems feasible with the advent of embedded web service technology.

It is clear that the step from Internet of Things towards a Web of Things will be taken sooner rather than later. The IoT can facilitate the realization of the WoT, opening up access to sensor data and stimulating their widespread usage, while at the same time avoiding vertically integrated and closed systems. As such, it presents great opportunities to researchers active as well in the field of web service technology as in the field of embedded distributed systems.

A.7.3 Interoperability

One of the key factors to ensure the widespread use of IoT devices is to support end-to-end interoperability between different devices. Currently interoperability between embedded devices is enforced by use of a standardized IETF LLN protocol stack. Despite the efforts of the IETF working groups, different interpretations

of the standards remain a threat that causes interoperability problems between implementations from different parties. Solving these issues requires identification of the possible problems and clarification of the implementation details where necessary to prevent possible ambiguities in the standards itself. Extensive interoperability testing events, like the ETSI Plugtests for CoAP and in the future possibility for 6LoWPAN, significantly help to improve the quality of the standards and the implementations.

Another more fundamental issue to consider is that of low-power interoperability, *i.e.*, interoperability when communicating parties employ (possibly different) MAC radio duty cycling techniques. The authors of [143] identify two open issues, namely that existing protocols for LLNs typically have not been designed for MAC duty cycling and that existing MAC duty cycling mechanisms have not been designed for interoperability. While most of the IETF work is situated above the MAC sublayer, it is important that these concerns are addressed in the future so that end-to-end IPv6 connectivity with embedded devices, that employ heavy MAC duty cycling, is possible.

A.7.4 Bringing semantics to the Web of Things

Semantics define a globally interpretable significance to data. Adding semantics to the IoT allows data that originates from different sources to be unambiguously accessible and processable across different domains and by different users. This allows describing data that is collected from the real world which helps automated processing and integration of said data into applications. Semantic descriptions are particularly useful in M2M environments where a high level of autonomy is assumed. Said descriptions can also help to facilitate discovery and management of IoT devices and their resources. In [144], the authors mention that the dynamicity and pervasiveness of the IoT domain poses additional challenges for semantic description when compared to conventional web service environments. More specifically, the volatility of the sensor data and the large scale of the IoT are new challenges when defining semantics when compared to traditional web services.

When looking at the resources on the devices themselves (e.g., a temperature) several resource representations can be explored: ranging from plain text formats, through formats defined by the IPSO alliance [145] to complex semantic representations using ontologies that are adapted to the specific applications and domains as described for instance in [146]. Furthermore, the SPITFIRE project [147] has defined vocabularies to describe sensors and to integrate them with W3C's Linked Open Data cloud [148]. This allows linking sensor data with other data that is already available on the World Wide Web (WWW). This brings the potential of semantic web technology (e.g., searching and reasoning) to constrained devices, realizing a Semantic Web of Things.

Similar to search engines in the WWW, sensors and their resources could be indexed just like regular web pages and made available to Internet users and other IoT devices. Of course, issues such as time dependent aspects should be taken into consideration (e.g., indexing a temperature sensor) introducing novel challenges and opportunities. Platforms like Cosm [149] already allow to make sensor data publicly available, browsable and searchable but lack the ability to actively crawl and index sensors.

A.7.5 Security and privacy in the Web of Things

Applications running on embedded devices and LLNs often require confidentiality and integrity protection. These security mechanisms can be provided at the application, transport, network, and/or at the link layer. In all these cases, prevailing constraints will influence the choice of a particular protocol. Some of the more relevant constraints are small code size, low power operation, low complexity, and small bandwidth requirements.

When securing embedded environments by using DTLS (as assumed within CoAP) at the transport layer, IPSec at the network layer, or reusing IEEE 802.15.4 security primitives within the 6LoWPAN adaptation layer; the problem of key distribution remains mostly untackled. It is often assumed that each device has an appropriate asymmetric public and/or symmetric key installed. How these keys actually are distributed and installed in a safe way, is often left open and is not part of current IETF standards. To help solve this issue, work is executed within IETF's *Smart Object Lifecycle Architecture for Constrained Environments* (SOLACE) [150].

Apart from enabling secure communications between parties, privacy of WoT users is also an important concern. As embedded devices measure and control the physical environment that surrounds a WoT user, they can be (ab)used to gain insight into a user's habits and whereabouts. Therefore it is vital to enforce adequate access control to this sensitive information, in order to protect the user's privacy. As more and more internet-enabled things will gain an online presence, transparent and easy-to-use management of what information devices may expose to the outside world will be needed. For instance, you might want to give your AC vendor access to the temperature sensors in your house without allowing them to turn on and off your lighting. While you might want to allow a different vendor to only control your lighting. One approach as discussed in [151] is to group embedded devices into virtual networks and only expose certain resources within each virtual network, thus achieving access control at the network layer. Another option is to provide access control at the application layer, where access to resources on embedded devices is granted/revoked on a per-vendor basis and the application itself enforces the access control.

A.7.6 Reprogrammability

Finally, many IoT devices will have a long lifetime and will be deployed at locations that are difficult to reach. As such, it is clear that some mechanism will be required to (wirelessly) update IoT devices with new or updated firmware, software, protocols and/or security keys. As an example, the DisSeNT project [152] already provides solutions to wirelessly add new application level components at run-time to embedded devices.

Providing wireless updates to IoT devices is at the moment quite difficult because it requires additional overhead in terms of the memory footprint of the software. In addition, (multi-hop) transmitting a firmware image to multiple embedded devices quickly depletes the batteries of involved devices. At the moment, very few embedded operating systems have the capability to execute these kinds of updates: additional research and standards will be needed before future-proof IoT networks can be deployed.

A.8 Conclusions

The popularity of sensor networks (and in a broader sense Internet of Things) has increased significantly over the last ten years. Integration of these embedded devices into the Internet is challenging, since they have characteristics that differ strongly from traditional internet devices, such as very limited energy, memory and processing capabilities. Initially, research focused on developing proprietary solutions that were typically vendor-specific and did not allow end-to-end connectivity between client devices and sensor devices. However, the use of standardized protocols enables the integration of constrained devices in the IPv6 Internet, both at the network level and at the service level.

In this paper, a high-level overview was given of the ongoing IETF standardization work that focuses on enabling direct connectivity between clients and sensor devices. To this end, different IETF groups are currently active. The IETF groups 6LoWPAN and ROLL focus on the IPv6 addressability and routing, whereas the IETF CoRE group focuses on realizing an embedded counterpart for RESTful web services. By combining these protocols, an embedded protocol stack can be created that has similar characteristics to traditional internet protocol stacks. In fact, the IETF protocols are designed to enable easy translation from internet protocols to sensor protocols and vice-versa.

The paper describes how the combination of IETF protocols enables flexible, direct interaction between internet clients and embedded Internet of Things devices. However, the paper also shows that the advent of standardized protocols is not an end point, but only a starting point for exploring additional open issues that should be solved to realize an all-encompassing Internet of Things. Several open

challenges remain such as resource representation, security, dealing with sleeping nodes, energy efficiency, integration with existing web service technologies and tools, linking with Cloud services, use of semantics, easy creation of applications, scalability, interoperability with other wireless standards, maintainability, *etc*.

Anyone involved in Internet of Things research (whether dealing with network layer aspects or service layer aspects) will, sooner or later, be confronted with the IETF protocols. This paper merely touches the surface of this broad domain and tries to encourage others to further explore the world of Internet-connected objects and tackle the mentioned remaining open issues and challenges.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 258885 (SPITFIRE project), from the iMinds ICON projects GreenWeCan and O'CareCloudS, a FWO postdoc grant for Eli De Poorter and a VLIR PhD scholarship to Isam Ishaq.

References

[1] N. Kushalnagar, G. Montenegro, and C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. RFC 4919 (Informational), August 2007. Available from: http://www.ietf.org/rfc/rfc4919.txt.

[2] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC 4944 (Proposed Standard), September 2007. Updated by RFCs 6282, 6775. Available from: http://www.ietf.org/rfc/rfc4944.txt.

[3] *Routing Over Low power and Lossy networks (roll)*, 2012. [Online; accessed 3 October 2012]. Available from: http://datatracker.ietf.org/wg/roll/.

[4] *ZigBee Alliance Plans Further Integration of Internet Protocol Standards*, 2012. [Online; accessed 3 October 2012]. Available from: https://docs.zigbee.org/zigbee-docs/dcn/09-5003.pdf.

[5] *Constrained RESTful Environments (core)*, 2012. [Online; accessed 28 December 2012]. Available from: http://datatracker.ietf.org/wg/core/.

[6] *IPv6 over Low power WPAN (6lowpan)*, 2012. [Online; accessed 28 December 2012]. Available from: http://datatracker.ietf.org/wg/6lowpan/.

[7] *IEEE 802.15.4*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.ieee802.org/15/pub/TG4.html.

[8] *IEEE Standard for Local and Metropolitan Area Networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2011.

[9] *ZigBee Alliance*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.zigbee.org/.

[10] *HART Communication Protocol and Foundation*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.hartcomm.org/.

[11] *MiWi Development Environment*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.microchip.com/miwi.

[12] *ISA100 Wireless Systems for Automation*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.isa.org/isa100.

[13] J.-P. Vasseur and A. Dunkels. *Interconnecting smart objects with ip: The next internet*. Morgan Kaufmann, 2010.

[14] J. Hui and P. Thubert. *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*. RFC 6282 (Proposed Standard), September 2011. Available from: http://www.ietf.org/rfc/rfc6282.txt.

[15] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*. RFC 6775 (Proposed Standard), November 2012. Available from: http://www.ietf.org/rfc/rfc6775.txt.

[16] E. Kim, D. Kaspar, and J. Vasseur. *Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*. RFC 6568 (Informational), April 2012. Available from: http://www.ietf.org/rfc/rfc6568.txt.

[17] E. Kim, D. Kaspar, C. Gomez, and C. Bormann. *Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing*. RFC 6606 (Informational), May 2012. Available from: http://www.ietf.org/rfc/rfc6606.txt.

[18] J. Nieminen, T. Savolainen, M. Isomaki, Z. Shelby, and C. Gomez. *Transmission of IPv6 Packets over BLUETOOTH Low Energy*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-6lowpan-btle-11.

[19] *IPv6 Over Low power WPAN (6LoWPAN) Charter*, 2012. [Online; accessed 13 December 2012]. Available from: http://datatracker.ietf.org/wg/6lowpan/charter/.

[20] J. Hui, D. Culler, and S. Chakrabarti. *6LoWPAN: Incorporating IEEE 802.15. 4 into the IP architecture*. IPSO Alliance White Paper, 3, 2009.

[21] G. Mulligan. *The 6LoWPAN architecture*. In Proceedings of the 4th workshop on Embedded networked sensors, pages 78–82. ACM, 2007.

[22] B. Cody-Kenny, D. Guerin, D. Ennis, R. Simon Carbajo, M. Huggard, and C. Mc Goldrick. *Performance evaluation of the 6LoWPAN protocol on MICAz and TelosB motes*. In Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, pages 25–30. ACM, 2009.

[23] M. R. Sulthana, P. Bhuvaneswari, and N. Rama. *Routing Protocols in 6LoWPAN: A Survey*. Eur. J. Sci. Res, 85(2):248–261, 2012.

[24] C. Borman. *6LoWPAN Generic Compression of Headers and Header-like Payloads*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-bormann-6lowpan-ghc-05.

[25] *Sahara Project*, 2012. [Online; accessed 10 December 2012]. Available from: http://sahara.tzi.org/.

[26] *HOBNET Project*, 2012. [Online; accessed 10 December 2012]. Available from: http://www.hobnet-project.eu/.

[27] *Outsmart: FP7 Framework Project*, 2012. [Online; accessed 10 December 2012]. Available from: http://www.fi-ppp-outsmart.eu/en-uk/Pages/default.aspx.

[28] D. Cassaniti. *A Multi-Hop 6LoWPAN Wireless Sensor Network for Waste Management Optimization*. 2011.

[29] *Calipso Project*, 2012. [Online; accessed 10 December 2012]. Available from: http://www.ict-calipso.eu/.

[30] R. Khoshdelniat. *6LoWPAN Applications and Internet of Things*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.apan.net/meetings/Hanoi2010/Session/SensNet.php.

[31] J. Schoenwaelder, A. Sehgal, T. Tsou, and C. Zhou. *Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-schoenw-6lowpan-mib-01.

[32] C. Bormann. *Guidance for Light-Weight Implementations of the Internet Protocol Suite.*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-lwig-guidance-02.

[33] V. Altmann, J. Skodzik, F. Golatowski, and D. Timmermann. *Investigation of the use of embedded web services in smart metering applications*. In IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, pages 6172–6177. IEEE, 2012.

[34] *Routing Over Low power and Lossy networks (roll)Charter*, 2012. [Online; accessed 27 December 2012]. Available from: http://datatracker.ietf.org/wg/roll/charter/.

[35] K. Pister, P. Thubert, S. Dwars, and T. Phinney. *Industrial Routing Requirements in Low-Power and Lossy Networks*. RFC 5673 (Informational), October 2009. Available from: http://www.ietf.org/rfc/rfc5673.txt.

[36] A. Brandt, J. Buron, and G. Porcu. *Home Automation Routing Requirements in Low-Power and Lossy Networks*. RFC 5826 (Informational), April 2010. Available from: http://www.ietf.org/rfc/rfc5826.txt.

[37] J. Martocci, P. D. Mil, N. Riou, and W. Vermeylen. *Building Automation Routing Requirements in Low-Power and Lossy Networks*. RFC 5867 (Informational), June 2010. Available from: http://www.ietf.org/rfc/rfc5867.txt.

[38] M. Dohler, T. Watteyne, T. Winter, and D. Barthel. *Routing Requirements for Urban Low-Power and Lossy Networks*. RFC 5548 (Informational), May 2009. Available from: http://www.ietf.org/rfc/rfc5548.txt.

[39] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550 (Proposed Standard), March 2012. Available from: http://www.ietf.org/rfc/rfc6550.txt.

[40] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel. *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*. RFC 6551 (Proposed Standard), March 2012. Available from: http://www.ietf.org/rfc/rfc6551.txt.

[41] P. Thubert. *Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)*. RFC 6552 (Proposed Standard), March 2012. Available from: http://www.ietf.org/rfc/rfc6552.txt.

[42] O. Gnawali and P. Levis. *The Minimum Rank with Hysteresis Objective Function*. RFC 6719 (Proposed Standard), September 2012. Available from: http://www.ietf.org/rfc/rfc6719.txt.

[43] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. *The Trickle Algorithm*. RFC 6206 (Proposed Standard), March 2011. Available from: http://www.ietf.org/rfc/rfc6206.txt.

[44] A. Conta, S. Deering, and M. Gupta. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. RFC 4443 (Draft Standard), March 2006. Updated by RFC 4884. Available from: http://www.ietf.org/rfc/rfc4443.txt.

[45] *TinyRPLTinyOS Documentation Wiki*, 2012. [Online; accessed 27 December 2012]. Available from: http://docs.tinyos.net/tinywiki/index.php/TinyRPL.

[46] N. Tsiftes, J. Eriksson, and A. Dunkels. *Low-power wireless IPv6 routing with ContikiRPL*. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pages 406–407. ACM, 2010.

[47] *Berkeleys OpenWSN Project*, 2012. [Online; accessed 27 December 2012]. Available from: http://openwsn.berkeley.edu/.

[48] *Nano-RK*, 2012. [Online; accessed 27 December 2012]. Available from: http://www.nanork.org/projects/nanork.

[49] J. Jeong, J. Kim, and P. Mah. *Design and implementation of low power wireless IPv6 routing for NanoQplus*. In Advanced Communication Technology (ICACT), 2011 13th International Conference on, pages 966–971. IEEE, 2011.

[50] B. Pavković, F. Theoleyre, and A. Duda. *Multipath opportunistic RPL routing over IEEE 802.15. 4*. In Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems, pages 179–186. ACM, 2011.

[51] L. B. Saad, C. Chauvenet, and B. Tourancheau. *Simulation of the RPL Routing Protocol for IPv6 Sensor Networks: two cases studies*. In International Conference on Sensor Technologies and Applications SENSORCOMM 2011. IARIA, 2011.

[52] *Contiki: The Open Source Operating System for the Internet of Things*, 2012. [Online; accessed 27 December 2012]. Available from: http://www.contiki-os.org/.

[53] L. Bartolozzi, T. Pecorella, and R. Fantacci. *ns-3 RPL module: IPv6 routing protocol for low power and lossy networks*. In Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques, pages 359–366. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012.

[54] S. K. Hammerseth. *Implementing RPL in a mobile and fixed wireless sensor network with OMNeT++*. 2011.

[55] *rpl-jsim-platformImplementation of RPL functionality in JSim platform-Google Project Hosting*, 2012. [Online; accessed 27 December 2012]. Available from: http://code.google.com/p/rpl-jsim-platform/.

[56] O. Gnawali and P. Levis. *Recommendations for Efficient Implementation of RPL*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-gnawali-roll-rpl-recommendations-04.

[57] J. Tripathi, J. de Oliveira, and J. Vasseur. *Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL)*. RFC 6687 (Informational), October 2012. Available from: http://www.ietf.org/rfc/rfc6687.txt.

[58] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler. *Contikirpl and tinyrpl: Happy together*. In Workshop on Extending the Internet to Low Power and Lossy Networks (IP+ SN), 2011.

[59] T. Clausen, U. Herberg, and M. Philipp. *A critical evaluation of the IPv6 routing protocol for low power and lossy networks (RPL)*. In Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on, pages 365–372. IEEE, 2011.

[60] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, and A. Durresi. *A performance analysis of point-to-point routing along a directed acyclic graph in low power and lossy networks*. In Network-Based Information Systems (NBiS), 2010 13th International Conference on, pages 111–116. IEEE, 2010.

[61] E. Baccelli, M. Philipp, and M. Goyal. *The p2p-rpl routing protocol for ipv6 sensor networks: Testbed experiments*. In Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on, pages 1–6. IEEE, 2011.

[62] J. Hui and R. Kelsey. *Multicast Protocol for Low power and Lossy Networks (MPL)*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-roll-trickle-mcast-02.

[63] G. Oikonomou and I. Phillips. *Stateless multicast forwarding with RPL in 6LowPAN sensor networks*. In Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on, pages 272–277. IEEE, 2012.

[64] S. Dawans, S. Duquennoy, and O. Bonaventure. *On link estimation in dense rpl deployments*. 2012.

[65] M. Goyal, E. Baccelli, A. Brandt, and J. Martocci. *A Mechanism to Measure the Routing Metrics along a Point-to-point Route in a Low Power and Lossy Network*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-roll-p2p-measurement-07.

[66] M. Goyal, E. Baccelli, M. Philipp, A. Brandt, and J. Martocci. *Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-roll-p2p-rpl-15.

[67] U. Herberg and T. Clausen. *A comparative performance study of the routing protocols LOAD and RPL with bi-directional traffic in low-power and*

lossy networks (LLN). In Proceedings of the 8th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, pages 73–80. ACM, 2011.

[68] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis. *Evaluating the Performance of RPL and 6LoWPAN in TinyOS*. In Workshop on Extending the Internet to Low Power and Lossy Networks (IP+ SN). Citeseer, 2011.

[69] N. Accettura, L. Grieco, G. Boggia, and P. Camarda. *Performance analysis of the RPL routing protocol*. In Mechatronics (ICM), 2011 IEEE International Conference on, pages 767–772. IEEE, 2011.

[70] N. Bressan, L. Bazzaco, N. Bui, P. Casari, L. Vangelista, and M. Zorzi. *The deployment of a smart monitoring system using wireless sensor and actuator networks*. In Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, pages 49–54. IEEE, 2010.

[71] Y. Chen, J.-P. Chanet, and K. M. Hou. *RPL Routing Protocol a case study: Precision agriculture*. In First China-France Workshop on Future Computing Technology (CF-WoFUCT 2012), pages 6–p, 2012.

[72] M. Becker, T. Pötsch, K. Kuladinithi, and C. Goerg. *Deployment of CoAP in Transport Logistics*. In Proceedings of the 36th IEEE Conference on Local Computer Networks (LCN). Bonn Germany 2011, 2011.

[73] J. Guo, P. Orlik, and G. Bhatti. *Loop Free DODAG Local Repair*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-guo-roll-loop-free-dodag-repair-00.

[74] J. Guo, P. Orlik, and G. Bhatti. *Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-guo-roll-loop-free-rpl-01.

[75] J. Ko, J. Jeong, J. Park, J. Jun, and N. Kim. *RPL Routing Pathology In a Network With a Mix of Nodes Operating in Storing and Non-Storing Modes*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ko-roll-mix-network-pathology-01.

[76] A. Baryun. *The Node Ability of Participation (NAP)*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-baryun-roll-nap-00.

[77] M. Goyal, D. Barthel, and E. Baccelli. *The Node Ability of Participation (NAP)*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-goyal-roll-dis-modifications-01.

[78] K.-S. Hong and L. Choi. *DAG-based multipath routing for mobile sensor networks*. In ICT Convergence (ICTC), 2011 International Conference on, pages 261–266. IEEE, 2011.

[79] K. C. Lee, R. Sudhaakar, J. Ning, L. Dai, S. Addepalli, J. Vasseur, and M. Gerla. *A comprehensive evaluation of rpl under mobility*. International Journal of Vehicular Technology, 2012, 2012.

[80] D. Carels, E. D. Poorter, I. Moerman, and P. Demeester. *Extending the IETF RPL routing protocol with mobility support*. preparation for Ad Hoc Networks journal, 2013.

[81] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. *Constrained Application Protocol (CoAP)*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-core-coap-13.

[82] W. Colitti, K. Steenhaut, and N. De Caro. *Integrating wireless sensor networks with the web*. Extending the Internet to Low power and Lossy Networks (IP+ SN 2011), 2011.

[83] A. Dunkels et al. *Efficient application integration in IP-based sensor networks*. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pages 43–48. ACM, 2009.

[84] Z. Shelby. *Embedded web services*. IEEE Wireless Communications, 17(6):52, 2010.

[85] Z. Shelby. *Constrained RESTful Environments (CoRE) Link Format*. RFC 6690 (Proposed Standard), August 2012. Available from: http://www.ietf.org/rfc/rfc6690.txt.

[86] C. Bormann and Z. Shelby. *Constrained Application Protocol (CoAP)*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-core-block-10.

[87] K. Hartke. *Observing Resources in CoAP*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-core-observe-07.

[88] *1st CoAP Plugtest*. Technical Report, April 2012.

[89] L. Velez. *IoT COAP2 Interop Event Preliminary Report*, 2012. [Online; accessed 28 December 2012]. Available from: http://svn.tools.ietf.org/svn/wg/core/Preliminary-Results-CoAP2.pdf.

[90] C. Lerche, K. Hartke, and M. Kovatsch. *Industry adoption of the internet of things: a constrained application protocol survey*. In Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on, pages 1–6. IEEE, 2012.

[91] A. P. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi. *Web Services for the Internet of Things through CoAP and EXI*. In Communications Workshops (ICC), 2011 IEEE International Conference on, pages 1–6. IEEE, 2011.

[92] *Californium (Cf) CoAP framework in Java*, 2012. [Online; accessed 28 December 2012]. Available from: http://people.inf.ethz.ch/mkovatsc/californium.php.

[93] *Copper (Cu) Add-ons for Firefox*, 2012. [Online; accessed 28 December 2012]. Available from: https://addons.mozilla.org/en-us/firefox/addon/copper-270430/.

[94] *Erbium (Er) REST Engine and CoAP Implementation for Contiki*, 2012. [Online; accessed 28 December 2012]. Available from: http://people.inf.ethz.ch/mkovatsc/erbium.php.

[95] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester. *Facilitating sensor deployment, discovery and resource access using embedded web services*. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, pages 717–724. IEEE, 2012.

[96] *evcoap*, 2012. [Online; accessed 28 December 2012]. Available from: https://github.com/koanlogic/webthings/tree/master/bridge/sw/lib/evcoap.

[97] *NanoService*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.sensinode.com/EN/products/nanoservice.html.

[98] *libcoap: C-Implementation of CoAP*, 2012. [Online; accessed 28 December 2012]. Available from: http://libcoap.sourceforge.net/.

[99] *CoAP. TinyOS Documentation Wiki*, 2012. [Online; accessed 28 December 2012]. Available from: http://docs.tinyos.net/tinywiki/index.php/CoAP.

[100] *coap.me*, 2012. [Online; accessed 28 December 2012]. Available from: http://coap.me/.

[101] *jcoap is a Java Library implementing the Constrained Application Protocol (CoAP)Google Project Hosting*, 2012. [Online; accessed 28 December 2012]. Available from: http://code.google.com/p/jcoap/.

[102] *Constraint Application Protocol (CoAP) for ERIKA embedded OS*, 2012. [Online; accessed 28 December 2012]. Available from: http://rtn.sssup.it/index.php/research-activities/middleware-of-things/middleware-of-things/11-research-activities/35-coaperika.

[103] *CoAPy: Constrained Application Protocol in PythonCoAPy v0.0.2 documentation*, 2012. [Online; accessed 28 December 2012]. Available from: http://coapy.sourceforge.net/.

[104] *ibr-alg/wiselib. GitHub*, 2012. [Online; accessed 28 December 2012]. Available from: https://github.com/ibr-alg/wiselib.

[105] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota. *Evaluation of constrained application protocol for wireless sensor networks*. In Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on, pages 1–6. IEEE, 2011.

[106] S. Duquennoy, N. Wirström, N. Tsiftes, and A. Dunkels. *Leveraging IP for Sensor Network Deployment*. In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+ SN 2011), Chicago, IL, USA, volume 11. Citeseer, 2011.

[107] T. Pötsch. *Performance of the Constrained Application Protocol for Wireless Sensor Networks*, 2011.

[108] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker, and C. Görg. *Implementation of coap and its application in transport logistics*. Proc. IP+ SN, Chicago, IL, USA, 2011.

[109] C. Bormann. *CoRE Roadmap and Implementation Guide*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-bormann-core-roadmap-03.

[110] Z. Shelby, S. Krco, and C. Bormann. *CoRE Resource Directory*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-shelby-core-resource-directory-04.

[111] A. Rahman and E. Dijk. *Group Communication for CoAP*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-ietf-core-groupcomm-04.

[112] C. Bormann. *CoAP Simple Congestion Control/Advanced*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-bormann-core-cocoa-00.

[113] A. Gurtov and E. Dashkova. *Computing the Retransmission Timeout in COAP*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.etsi.org/plugtests/COAP2/Presentations/08_Computing_Retransmission_Timeout.pdf.

[114] B. Greevenbosch. *CoAP Minimum Request Interval*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-greevenbosch-core-minimum-request-interval-00.

[115] S. Li, J. Hoebeke, and A. Jara. *Conditional observe in CoAP*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-li-core-conditional-observe-02.

[116] G. Ketema, J. Hoebeke, I. Moerman, P. Demeester, L. S. Tao, and A. J. Jara. *Efficiently observing Internet of Things Resources*. In Green Computing and Communications (GreenCom), 2012 IEEE International Conference on, pages 446–449. IEEE, 2012.

[117] M. Vial. *CoRE Mirror Server*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-vial-core-mirror-server-00.

[118] J. Hoebeke, D. Carels, I. Ishaq, G. Ketema, J. Rossey, E. Depoorter, I. Moerman, and P. Demeester. *Leveraging upon standards to build the Internet of Things*. In Communications and Vehicular Technology in the Benelux (SCVT), 2012 IEEE 19th Symposium on, pages 1–6. IEEE, 2012.

[119] A. Rahman. *Enhanced Sleepy Node Support for CoAP*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-rahman-core-sleepy-01.

[120] *Application Layer Protocol Support for Sleeping Nodes in Constrained Networks. US Patent 20120151028*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.google.com/patents/US20120151028.

[121] H. Tschofenig. *Report from the Smart Object Security Workshop*, 2012. [Online; accessed 18 December 2012]. Available from: http://www.ietf.org/proceedings/83/slides/slides-83-saag-3.pdf.

[122] O. Garcia-Morchon, S. Keoh, S. Kumar, R. Hummen, and R. Struik. *Security Considerations in the IP-based Internet of Things*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-garcia-core-security-04.

[123] O. Bergmann, S. Gerdes, S. Schafer, F. Junge, and C. Bormann. *Secure bootstrapping of nodes in a CoAP network*. In Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE, pages 220–225. IEEE, 2012.

[124] K. Hartke and O. Bergmann. *Datagram Transport Layer Security in Constrained Environments*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-hartke-core-codtls-02.

[125] S. Raza, D. Trabalza, and T. Voigt. *6LoWPAN compressed DTLS for CoAP*. In Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on, pages 287–289. IEEE, 2012.

[126] H. Tschofenig and J. Gilger. *A Minimal (Datagram) Transport Layer Security Implementation*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-tschofenig-lwig-tls-minimal-012.

[127] M. Brachmann, O. Garcia-Morchon, and M. Kirsche. *Security for practical coap applications: Issues and solution approaches*. In the 10th GI/ITG KuVS Fachgespraech Sensornetze (FGSN11), 2011.

[128] T. Kivinen. *Minimal IKEv2*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-kivinen-ipsecme-ikev2-minimal-01.

[129] A. Castellani, S. Loreto, A. Rahman, T. Fossati, and E. Dijk. *Best Practices for HTTP-CoAP Mapping Implementation*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-castellani-core-http-mapping-05.

[130] A. Castellani, S. Loreto, A. Rahman, T. Fossati, and E. Dijk. *Best Practices for HTTP-CoAP Mapping Implementation*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-castellani-core-advanced-http-mapping-00.

[131] M. Becker, K. Li, K. Kuladinithi, and T. Poetsch. *Transport of CoAP over SMS, USSD and GPRS*, 2012. [Online; accessed 28 December 2012]. Available from: http://tools.ietf.org/html/draft-becker-core-coap-sms-gprs-02.

[132] M. Kovatsch, S. Mayer, and B. Ostermaier. *Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things*. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, pages 751–756. IEEE, 2012.

[133] S. Hans. *Secure Environment management based on CoAP*. In Proceedings of the ETSI CoAP Workshop; Sophia Antipolis, France, pages 27–30, 2012.

[134] M. Castro, A. J. Jara, and A. Skarmeta. *Architecture for improving terrestrial logistics based on the web of things*. Sensors, 12(5):6538–6575, 2012.

[135] J. Mäenpää, J. J. Bolonio, and S. Loreto. *Using RELOAD and CoAP for wide area sensor and actuator networking*. EURASIP Journal on Wireless Communications and Networking, 2012(1):1–22, 2012.

[136] A. Rahman, D. Gellert, and D. Seed. *A gateway architecture for interconnecting smart objects to the internet*. In Proceedings of the Workshop Interconnecting Smart Objects with the Internet, Prague, Czech Republic, volume 25, 2011.

[137] B. C. Villaverde, D. Pesch, R. De Paz Alberola, S. Fedor, and M. Boubekeur. *Constrained application protocol for low power embedded networks: A survey*. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, pages 702–707. IEEE, 2012.

[138] D. Barbieri. *CoAP improvements to meet embedded device hardware constraints Workshop Prague, 25 th March 2011*.

[139] H. Yahyaoui, Z. Maamar, and K. Boukadi. *A framework to coordinate web services in composition scenarios*. International Journal of Web and Grid Services, 6(2):95–123, 2010.

[140] L. Gao, S. D. Urban, and J. Ramachandran. *A survey of transactional issues for web service composition and recovery*. International Journal of Web and Grid Services, 7(4):331–356, 2011.

[141] W. Kim. *Cloud computing adoption*. International Journal of Web and Grid Services, 7(3):225–245, 2011.

[142] J. M. Rodriguez, A. Zunino, and M. Campo. *Introducing mobile devices into grid systems: a survey*. International Journal of Web and Grid Services, 7(1):1–40, 2011.

[143] A. Dunkels, J. Eriksson, and N. Tsiftes. *Low-power Interoperability for the IPv6-based Internet of Things*. In Proceedings of the 10th Scandinavian Workshop on Wireless Ad-Hoc Networks (ADHOC11), Stockholm, Sweden, pages 10–11, 2011.

[144] P. Barnaghi, W. Wang, C. Henson, and K. Taylor. *Semantics for the Internet of Things: early progress and back to the future*. International Journal on Semantic Web and Information Systems (IJSWIS), 8(1):1–21, 2012.

[145] Z. Shelby and C. Chauvenet. *The IPSO Application Framework*, 2012. [Online; accessed 28 December 2012]. Available from: www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf.

[146] B. Abdulrazak, B. Chikhaoui, C. Gouin-Vallerand, and B. Fraikin. *A standard ontology for smart spaces*. International Journal of Web and Grid Services, 6(3):244–268, 2010.

[147] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth, et al. *SPITFIRE: toward a semantic web of things*. Communications Magazine, IEEE, 49(11):40–48, 2011.

[148] *SweoIG/TaskForces/CommunityProjects/LinkingOpenData. W3C Wiki*, 2012. [Online; accessed 28 December 2012]. Available from: http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData.

[149] *Cosm. Internet of Things Platform Connecting Devices and Apps for Real-Time Control and Data Storage*, 2012. [Online; accessed 28 December 2012]. Available from: https://cosm.com/.

[150] *solace Info Page*, 2012. [Online; accessed 28 December 2012]. Available from: https://www.ietf.org/mailman/listinfo/solace.

[151] I. Ishaq, J. Hoebeke, I. Moerman, and P. Demeester. *Internet of Things virtual networks: Bringing network virtualization to resource-constrained devices*. In Green Computing and Communications (GreenCom), 2012 IEEE International Conference on, pages 293–300. IEEE, 2012.

[152] *The DisSeNT project*, 2012. [Online; accessed 10 December 2012]. Available from: http://distrinet.cs.kuleuven.be/software/dissent/.

B

Internet of Things Virtual Networks: Bringing Network Virtualization to Resource-constrained Devices

The main focus of this dissertation is on creating enablers for the Internet of Things (IoT) using the Constrained Application Protocol (CoAP). We have focused on the aspects of CoAP that are related to discovering resources and grouping them in way that facilitates their usage efficiently. In this Appendix, we are still dealing with constrained devices and environments. However, we present an alternative approach to group constrained devices using virtualization techniques. This approach focuses on the objects, both resource-constrained and non-constrained, that need to cooperate by integrating them into a secured virtual network, named an Internet of Things Virtual Network or IoT-VN. Inside this IoT-VN full end-to-end communication can take place through the use of protocols that take the limitations of the most resource-constrained devices into account, such as CoAP.

⋆ ⋆ ⋆

I. Ishaq, J. Hoebeke, I. Moerman, and P. Demeester.

Published in proceedings of the 2012 IEEE International conference on cyber, physical and social computing, Nov. 2012.

Abstract Networks of smart resource-constrained objects, such as sensors and actuators, can support a wide range of application domains. In most cases these networks were proprietary and stand-alone. More recently, many efforts have been undertaken to connect these networks to the Internet using standard protocols. Current solutions that integrate smart resource-constrained objects into the Internet are mostly gateway-based. In these solutions, security, firewalling, protocol translations and intelligence are implemented by gateways at the border of the Internet and the resource-constrained networks. In this paper, we introduce a complementary approach to facilitate the realization of what is called the Internet of Things. Our approach focuses on the objects, both resource-constrained and non-constrained, that need to cooperate by integrating them into a secured virtual network, named an Internet of Things Virtual Network or IoT-VN. Inside this IoT-VN full end-to-end communication can take place through the use of protocols that take the limitations of the most resource-constrained devices into account. We describe how this concept maps to several generic use cases and, as such, can constitute a valid alternative approach for supporting selected applications. A first implementation demonstrating the key concepts of this approach is described. It illustrates the feasibility of integrating resource-constrained devices into virtual networks, but also reveals open challenges.

B.1 Introduction

The Internet and its applications are evolving in many directions. In one direction, it is expected that small, embedded devices such as sensors and actuators will become a cornerstone of the Future Internet transforming it into the IoT. These devices enable us to collect information about the physical world and inject it into the virtual world where it can be used for a plethora of applications, crossing many application domains. To realize this Internet of Things and to be able to make use of the data generated by these resource-constrained devices, these devices need to be integrated into the Internet and to be able to connect to other devices. They have to make their data accessible to interested parties, which can be web services, smart phones, cloud resources, *etc*.

Making these data available through the Internet is one thing; doing this in a controlled way, not exposing data to the whole world, is another thing. As such, integrating resource-constrained devices into the Internet is more than simply connecting these devices to the Internet in one way or another. The traditional approach to achieve this, is through the use of gateways that reside at the border of the Internet and the sensor networks. They incorporate intelligence and access control, collect the sensor data themselves and expose it to the Internet. This approach has certain advantages, but also limitations. In some cases, interested parties want to be able to access the data directly, requiring direct connectivity to

the sensors. Here, the gateways play a less prominent role, primarily dealing with the translation of Internet protocols to sensor protocols and vice versa, implying that more complexity is shifted to the sensors.

In this paper we present a complimentary approach, by integrating all objects that need to cooperate, including both resource-constrained and non-constrained devices, into a secured virtual network, called an Internet of Things Virtual Network or IoT-VN, in which direct end-to-end communication can take place. This approach was named Managed Ecosystems of Networked Objects (MENO) and was first introduced and discussed in detail in [1]. Such an ecosystem was defined as "*a completely independent, managed, observable, virtual environment of interdependent, networked objects that cooperate in harmony*." In this paper we now lay the foundation of this new concept by presenting a first implementation demonstrating its key concepts. As such, the main contributions of this paper are the following:

- A detailed discussion of the IoT-VN concept and its potential benefits in several use cases.
- A middleware for non-constrained devices to securely exchange raw data over layer 2 or layer 3 virtual links inside a self-organized virtual network.
- Simple extension to resource-constrained devices: using neighbor discovery direct virtual links can be established.
- An evaluation of the feasibility to run proprietary lightweight protocols inside the resulting virtual network.

In the following, we first describe in more detail the current approaches to integrate resource-constrained devices into the Internet and their advantages and limitations (Section B.2). Next, we briefly describe our approach and compare it with related work (Section B.3), followed by an illustration of how it can be applied to a number of generic use cases (Section B.4). Our implementation and first test deployment are then described in Sections B.5 and B.6. Finally we summarize our conclusions in Section B.7.

B.2 Current approaches and limitations

It is envisioned that resource-constrained devices, such as sensors and actuators will play an important role in the Future Internet and will enable a whole new set of novel services. These devices are considered resource-constrained because they have less memory, CPU, energy and bandwidth than typical hosts. To preserve energy, these devices often have long sleeping schemas and they typically form networks that are lossy and are a lot less reliable than Ethernet. For these reasons

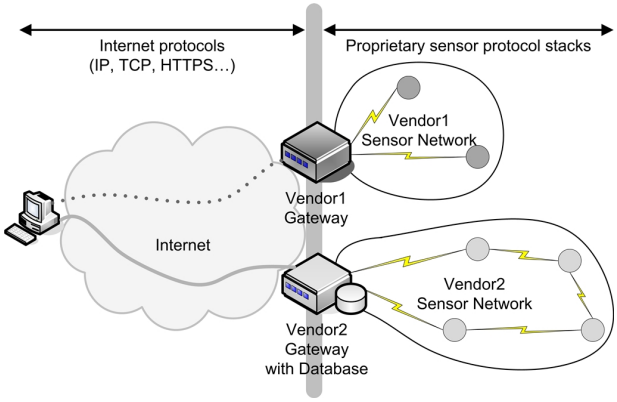


Figure B.1: Gateways are often used to interconnect sensor networks to the Internet.

current Internet Protocols are generally considered not suitable to run on resource-constrained devices. Other approaches are required to expose services offered by these resource-constrained devices to the outside world. Most of these approaches can be categorized into two main categories: use of gateways and integration of sensors into the IP-world.

B.2.1 Use of gateways

A multitude of specialized control protocols for resource-constrained devices are now used in the industry, e.g. the ZigBee[1] standard or BACnet[2] - a data communication protocol for building automation and control networks. In the absence of widely accepted standard protocols, many vendors were encouraged to develop proprietary protocols to run inside their sensor networks.

Connectivity between the Internet and many of the sensor networks is nowadays achieved through the use of gateways or proxies. These gateways have to translate between protocols used in the Internet and protocols used in the sensor networks.

Figure B.1 displays two various sensor networks that are connected to the Internet by gateways. Users on the Internet have to connect to the gateways in order to obtain data from the respective sensor network. There are several ways how a gateway can handle such user requests.

- The gateway from vendor1 translates standard Internet protocols into the proprietary sensor protocols and relays the requests to the sensors in its network. It then gets the answers from the relevant sensors by means of the

[1] http://www.zigbee.org/
[2] http://www.bacnet.org/

proprietary sensor protocols and sends back the appropriate replies to the user using standard Internet protocols. The gateway offers an API that applications should use in order to create requests that can be understood by the gateway.

- Although it might look the same to the user, the gateway from vendor2 behaves in a different way than the gateway from vendor1. This gateway contains a database with pre-collected sensor data. When it gets a request from a user on the Internet, it replies directly to the requester using the data in the database. In some cases, the gateway is simply running a web server that makes the data available to the outside world. The user usually cannot tell, whether the returned data is coming in real-time from the sensors or whether it is coming from a value that has been previously stored in a database.

The use of gateways has certainly many advantages. Since it provides a single entry to the sensor network, it shields the sensor resources from the Internet and enables a high degree of access control. If needed, the gateway can also process and aggregate data from different sensors and present them in a uniform way to the users.

However there are also disadvantages that accompany the use of gateways. The gateway is the only entity that can talk to the sensors directly. Due to the lack of real end-to-end connectivity or interaction with the resource-constrained devices, flexibility of usage is reduced. Users can query the sensors only in the way that is allowed by the gateway. Adding new sensor resource often requires adaptation on the gateway. Another disadvantage is the vendor lock-in. Gateway and sensors often have to be from the same vendor in order to be compatible.

B.2.2 Integration of sensors into the IP-world

To address the networking needs of resource-constrained devices the IETF has formed several working groups: IPv6 over Low Power WPAN (6LoWPAN) [2], Routing Over Low Power and Lossy Networks (ROLL) [3], Constrained Restful Environments (CORE) [4], and Constrained Application Protocol (CoAP) [5].

Similar to the previous category of approaches, gateways are still used to translate between protocols used in the Internet and protocols used in the sensor networks, e.g. IPv6 to 6LoWPAN and vice versa (Figure B.2). However the difference here is that through the use of standard protocols, many of the disadvantages from the previous approaches are now taken care of. For example it is now possible to have the gateway and the sensors from different vendors.

Flexibility is also improved by this approach. Users can now query the sensors without the need for the gateway to understand the query and the data itself. The application payload can now travel directly from the client to the sensor, where it is

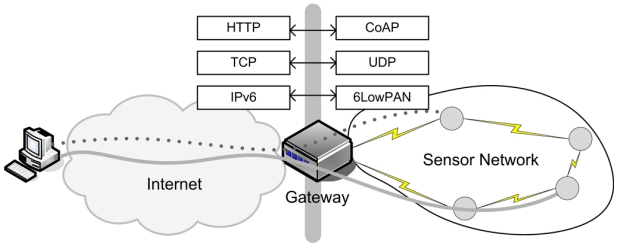


Figure B.2: Internet protocols are extended to the sensor networks. The Gateway translates between the two protocol stacks.

processed and acted upon. The gateway takes care of the translation between standardized protocols. This makes adding and removing sensor resources transparent to the gateway and improves interoperability of devices.

Of course, by allowing direct interaction with the resource-constrained devices, new challenges related to access control, authentication, *etc*. are introduced, up to the level where the resource-constrained devices themselves need to manage access to the resources they offer. These security aspects are considered a major challenge in the IP-based Internet of Things [6]. In summary, both approaches have their advantages and disadvantages, characterized by the degree of openness in accessing the services on the resource-constrained devices. In the next section, we present a third, novel, complementary approach.

B.3 IoT-VN

In several use cases, there is no need to expose the data generated by resource-constrained devices to the whole world. Only a limited number of devices are involved, both resource-constrained and non-constrained, that need to cooperate in order to achieve a specific goal. Based on this observation, we propose adding a complementary approach to the approaches targeting such use cases. This approach was first introduced and discussed in [1] and named "managed ecosystem of networked objects" (MENO). It aims to realize a secured and confined environment in which all objects that need to cooperate can communicate in an end-to-end manner as shown in Figure B.3. This is achieved by creating a virtual network of all involved devices, including resource-constrained devices. In the remainder of this paper we refer to this virtual network as an Internet of Things Virtual Network or IoT-VN. An IoT-VN is established on top of different types of physical networks and consists of virtual links. A virtual link can be established between two devices connected to the Internet (on top of layer 3 IP connectivity), between two devices in a LAN (on top of layer 2 connectivity) or between two devices in a resource-constrained network such as a sensor network. Inside this virtual network, end-to-

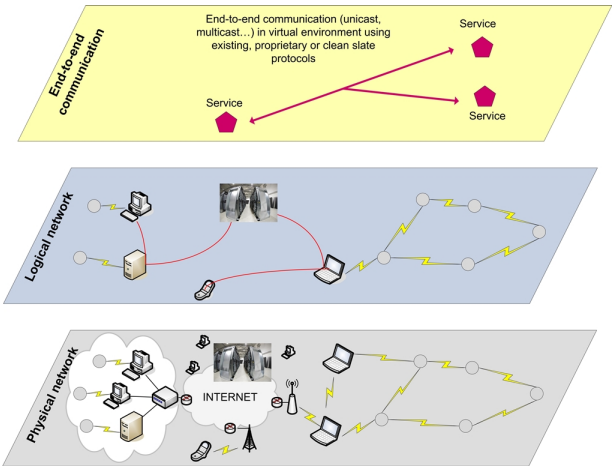


Figure B.3: MENO Concept – IoT-VN approach.

end communication between any two devices or groups of devices can take place through the use of existing protocols for communication in resource-constrained networks, through the use of proprietary protocols or through the use of novel protocols specifically designed for the targeted use case. Applications or services running inside the virtual network only see this logical layer. In a transparent way, all protocols running inside the virtual network communicate by transmitting their data over the virtual links. The logic for managing and establishing the virtual network automatically takes care of connecting the different members of the virtual network through the establishment of the virtual links and the secure transmission of data over these links. As such, security is inherently part of the design at the connectivity level already, by only allowing access to devices that are member of the virtual network. For more details about the concept, we refer to [1].

To the best of our knowledge, creating virtual networks among resource-constrained and non-constrained devices in the same setting has not received much attention in the literature so far. Still, it has been addressed a few times in the past years. According to the VITRO/FP7 project [7], Virtual Sensor Networking is an emergent approach which enables the dynamic collaboration of a subset of sensor nodes, not necessarily controlled or owned by the same Administrative Domain, aiming to complete a certain task or computation at a given time.

Reference [8] considers collaboration and resource sharing to be the main idea of Virtual Sensor Networks. To achieve this, nodes can be grouped into different Virtual Sensor Networks based on the phenomenon they track or the task they perform. Virtual Sensor Networks are expected to provide the protocol support for formation, usage, adaptation, and maintenance of subset of sensors collaborating on a specific task. Furthermore, Virtual Sensor Networks should make efficient

use of intermediate nodes, networks, or other Virtual Sensor Networks to deliver messages across members of a Virtual Sensor Network. On the other hand, virtual networking is common in the Internet world such as VLAN [9], VPN [10], or VPAN [11]. Although network virtualization is frequently used to achieve secure communication over the unsecure Internet, the integration of resource-constrained devices is largely unexplored. End-to-End secure communication between IP-enabled resource-constrained devices and the traditional Internet using IPsec was demonstrated in [12]. This approach provides secure communication on the network layer between end points and the technology presented there could be used to establish tunnels with resource-constrained devices, if complemented with appropriate solutions to establish trust between all IoT-VN members. However, the IoT-VN approach also aims to include layer 2 secure communication between neighboring devices and aims to realize a complete virtual network involving resource-constrained and non-constrained devices. Further, we do not limit ourselves to secure IP communication, but explore the possibilities of running protocols adapted to the limitations of resource-constrained devices inside the virtual environment. When considering the realization of trust, IoT-VNs can also benefit from ongoing research on bringing security solutions in reach of resource-constrained devices such as [13] and [14].

To the best of our knowledge, no other related work has taken our approach in providing a flexible virtualized network that contains both resource-constrained and non-constrained devices in the same virtual network.

B.4 Example use cases

IoT-VNs can be beneficial in several use cases, when compared to the traditional integration approaches described in Section II. In this section we explore several generic use cases of IoT-VNs.

B.4.1 Partitioning a sensor network

The simplest application of IoT-VNs is the partitioning of a sensor network into two or more virtual networks. In Figure B.4 a virtual network is shown that only contains a subset of the available sensors in a sensor network. Secure communication is available only between the members of the virtual network. Other sensors of the sensor network may still be used to forward traffic between the IoT-VN members, however these sensors will not be able to interpret the data being forwarded.

This can be used for example in building management where sensor networks belonging to different administrators, owners, departments, *etc.* are deployed. These networks cooperate to enhance data forwarding, but all internal traffic for control, management and data collection is shielded.

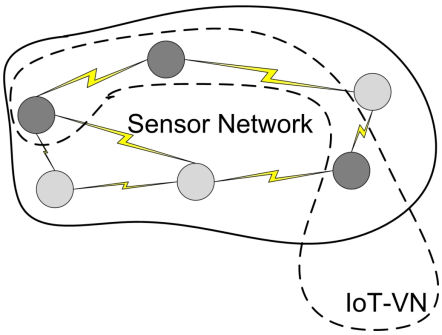


Figure B.4: An IoT-VN that only contains a subset of the available sensors in a sensor network. Secure communication is available only between members of the virtual network.

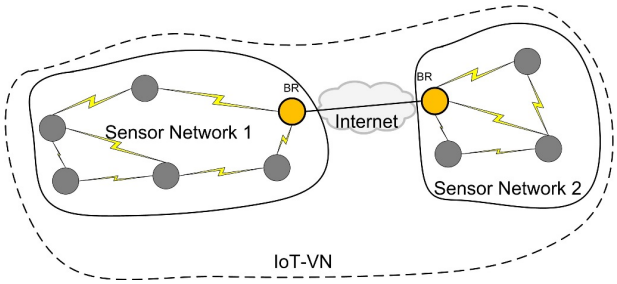


Figure B.5: An IoT-VN that combines several sensor networks into one big virtual sensor network, by establishing a virtual link between the two Border Routers (BR). Secure communication is available between all members of the IoT-VN, regardless of their physical location.

B.4.2 Aggregating multiple sensor networks

It can be required to connect two or more geographically separate sensor networks. In Figure B.5 an IoT-VN that combines multiple sensor networks into a single big virtual sensor network is shown. The two networks can be combined for example by creating a secure Layer-3 tunnel over the Internet between the two Border Routers (BR) of the sensor networks. Secure communication is available between all the members of the virtual network, regardless of their physical location and connection to the network. Typical sensor network protocols running inside the IoT-VN only see a single large sensor network.

This can be used for example in case sensors or actuators in one sensor network should directly act upon measurements collected in another, not directly connected, sensor network.

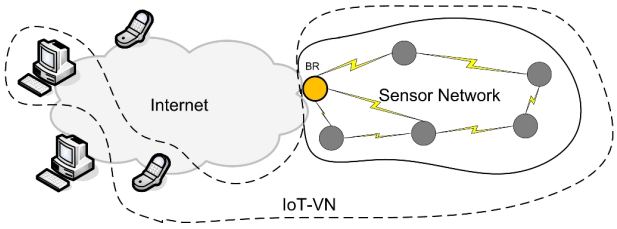


Figure B.6: An IoT-VN that is extended to include non-constrained devices.

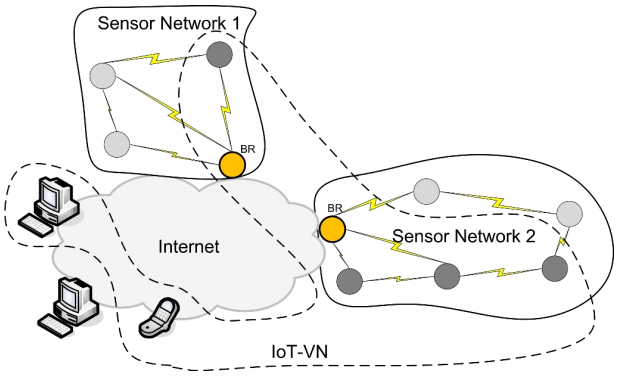


Figure B.7: A hybrid IoT-VN.

B.4.3 Extending a sensor network with non-constrained devices

In many cases, it is required to have one or more non-constrained devices as part of the sensor network. These devices are used for example for storage or in order to perform calculations that are beyond the capabilities of the sensors. Figure B.6 shows an IoT-VN that is extended to include remote non-constrained devices.

For example, a server in the cloud can be used to directly gather information from all sensor nodes in the IoT-VN, aggregate that information, and present it to the Internet in a controlled and secure manner. The cloud server transparently acts as the sink of the network and can run the corresponding protocols.

B.4.4 A hybrid of sensor networks

Of course, all of the above scenarios can be combined together as needed to fit the needs of the network owner. In Figure B.7 an IoT-VN is shown, that is a hybrid of the previous scenarios. This IoT-VN is created by combining partitions from two separate sensor networks and extending them with remote non-constrained devices. It is clear that the number of possible configurations is almost unlimited and will be strongly dependent on the use case that needs to be realized.

B.5 Implementation

The goal of our implementation is the establishment of an IoT-VN, a secure self-organizing virtual network environment. The virtual network is a collection of virtual links on top of which data can be exchanged and new protocols can be designed from scratch. It should be possible that devices of various capabilities are able to be part of the IoT-VN. To that end, two separate but interoperable implementations were developed. The first implementation is designed for non-constrained devices running typical operating systems such as Linux, Windows, OS-X, embedded Linux, *etc*. The second implementation targets resource-constrained devices such as sensors that run specific operating systems.

B.5.1 Non-constrained implementation

The non-constrained implementation has been realized in Click Router, a C++ based modular framework that can be used to realize any network packet processing functionality [15]. It consists of several modules that perform a specific task. The modules are combined together using a configuration file to obtain the overall functionality of the system.

As already mentioned, nodes in the IoT-VN can be thought of as being connected by virtual or logical links. These virtual links correspond to a path in the underlying network, perhaps through multiple physical links. For non-constrained devices these virtual link are established either over available layer 3 IP connectivity (e.g. over the Internet) or directly over available layer 2 LAN connectivity (when in the same broadcast domain). Since not every device is allowed to participate in the IoT-VN, a mechanism is needed to identify the members of the IoT-VN. Therefore all devices participating in the IoT-VN share a common cryptographic trust relationship consisting of a public and private key pair, signed by a common IoT-VN private key maintained by a certification authority (implemented using OpenSSL).

When members are in the same broadcast domain (e.g. Ethernet, Wi-Fi), they can discover each other by periodically sending beacon packets. Upon the reception of such a beacon, a challenge-response mechanism is initiated to authenticate each other and negotiate a symmetric session key resulting in a secure virtual link between both nodes.

When members are connected to the Internet (via cable, Wi-Fi, UMTS, GPRS, *etc*.) they can register with a trusted agent. This information is then exchanged with already registered members and subsequently used to establish tunnels between the different members. The implementation includes NA(P)T detection, hole punching and relaying via the trusted agent in order to deal with NA(P)T boxes.

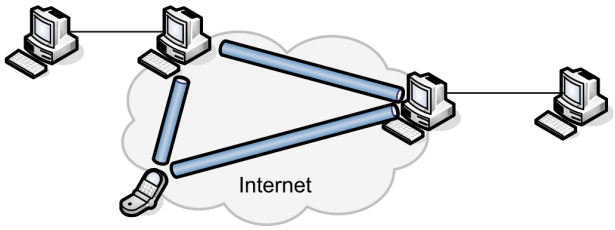


Figure B.8: An IoT-VN that contains layer 2 (same broadcast domain) and layer 3 (IP connectivity over Internet) virtual links.

Using these neighbor detection and tunneling mechanisms, layer 2 and layer 3 virtual links can be established as illustrated in Figure B.8. A Virtual Link Manager module manages all virtual links. Together with a Virtual Link Forwarder module, it is possible to send any data over a single virtual link or over multiple virtual links. Also, upon reception of data over a virtual link, the virtual link over which the data arrived is identified. Using this basic functionality it is possible to deploy any protocol on top of the established virtual links.

Further some modules are present to maintain and manage the IoT-VN and to deal with all used network interfaces and changes in their properties. Part of this implementation is based on the VPAN implementation described in [11], but was thoroughly modified in order to become IP agnostic.

B.5.2 Extension of the IoT-VN concept to resource-constrained devices

The extension of IoT-VNs to sensors consists of three parts, which will be further described in this section: an extension of the non-constrained implementation to use a physical network interface to the sensor network (802.15.4 network interface), an implementation of the IoT-VN concept on resource-constrained devices and new modules for the non-constrained implementation for establishing virtual links with a resource-constrained devices over the 802.15.4 interface.

Although the non-constrained implementation mentioned in section A runs well on devices down to the level of a smart objects with embedded Linux, this implementation has a footprint that is way beyond the capabilities of typical resource-constrained devices (e.g., 48K bytes of ROM, 8K bytes of RAM). In order to have an implementation that would run on such devices, customized operating systems and development environment should be used. These environments try to optimize the footprint of the generated code as much as possible, in order for it to fit on the limited resource of today's typical resource-constrained devices.

Our resource-constrained devices implementation of the IoT-VN has been realized using the IDRA framework [16]. IDRA is a network architecture and appli-

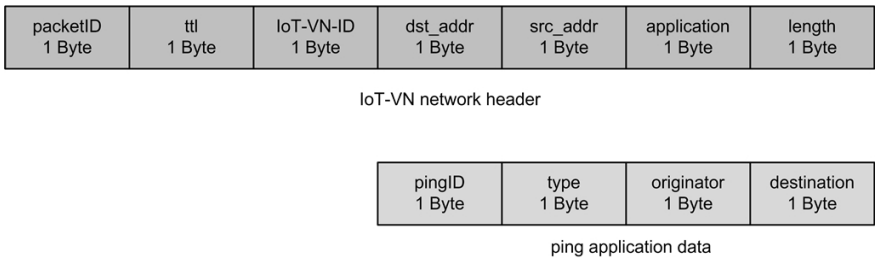


Figure B.9: Top: IoT-VN Network header format. Bottom: ping application data format.

cation platform written in nesC and developed for TinyOS [17] - an event-driven operating system designed for sensor network nodes that have very limited resources.

This implementation uses a similar design and the same packet format (e.g. beacon format) as the non-constrained implementation and is thus compatible with it. However, this implementation does not include all the features that are supported in non constrained implementation (e.g. data encryption).

In addition to the previous described two types of virtual links, a third type has been introduced. This type is used whenever at least one node of the virtual link nodes is a resource-constrained device. In this way it becomes possible for a non-constrained device to adopt its communication to the limitations imposed be the resource-constrained device. For example it would be possible to use a more light-weight (but maybe a weaker) encryption algorithm in the communication between a resource-constrained and a non-constrained device.

B.5.3 Protocols inside the IoT-VN

Inside the IoT-VN it is possible to run either standard sensor protocols or to design and run completely new protocols that fit the needs of the particular IoT-VN. In order to demonstrate both cases we have implemented two simple protocols: Ad hoc On-Demand Distance Vector (AODV) Routing [18] and a PING application.

B.5.3.1 AODV

To enable multi-hop communication between nodes, we implemented the basic functionality of the existing AODV protocol. To achieve this, every member of the IoT-VN is assigned a unique 1-byte address. Also a proprietary network header as shown in the top part of Figure B.9 is defined. For AODV routing messages, the AODV header is appended to this network header. When one member, the source, wants to transmit data to another member, the destination, and no route has been established yet, a route request is broadcasted inside the IoT-VN. The source sends

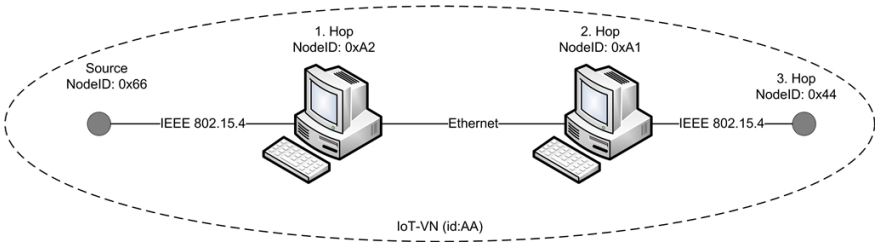


Figure B.10: The network that was used to test the IoT-VN implementation on PCs running Linux and on sensors.

the route request over all established virtual links using the virtual link broadcasting functionality. All receiving nodes repeat this process until the destination is reached. This establishes a reverse path to the source node, specifying the next hop 1-byte address and the virtual link over which the data needs to be sent to reach the next hop towards the source. Upon reception of the route request by the destination, the destination will send back a route reply using the backward path to the source. While the reply travels from the destination to the source, a forward path is established. This completes the establishment of the route.

B.5.3.2 Ping

A simple ping application has been implemented, consisting of a ping request and a ping reply. The header of this proprietary ping protocol is shown at the bottom part of Figure B.9. This header is appended to the network header, with the application field set to a value that is assigned to the ping protocol. The ping packets are forwarded using the routes established by AODV.

Using AODV and Ping it is possible to demonstrate communication between any two members inside the IoT-VN. A well-known protocol, AODV, is used, but complemented with other things such as the 1-byte address, network header, *etc*. which are only known inside this IoT-VN. Together it realizes end-to-end communication capabilities inside the IoT-VN, capable of also running on resource-constrained devices.

B.6 Results

In order to test our resource-constrained and non-constrained implementations and the interoperability between these two implementations, we built a small network that consists of two sensors and two PCs as shown in Figure B.10. Each of the sensors was connected to one PC wirelessly using IEEE 802.15.4. The two PCs were connected together via an Ethernet link. All four devices were manually

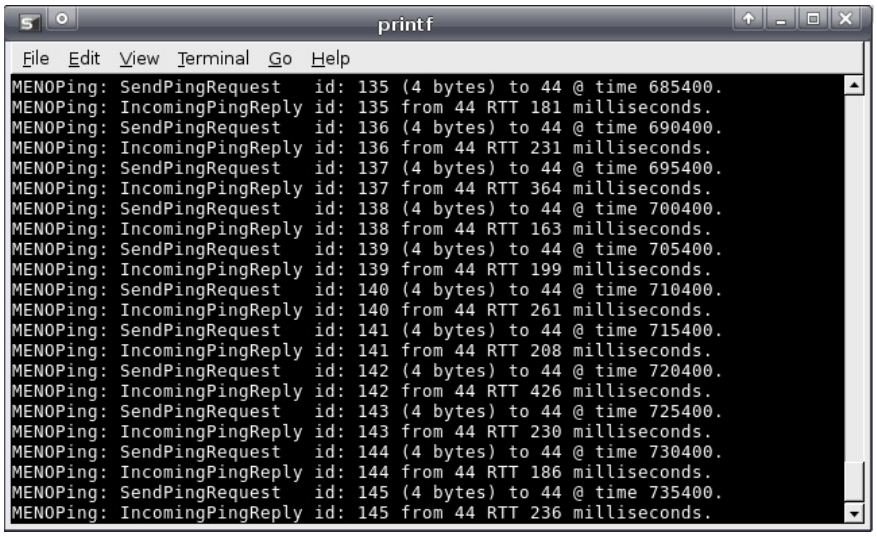


Figure B.11: Screenshot from the ping application showing successful ping between two sensor that belong to the same IoT-VN. The address of the destination is the nodeID in the virtual network. The output was taken by connecting to the USB interface of the sending sensor.

configured to be part of the same IoT-VN – identified by ID AA.

When the applications run on the devices, neighboring devices discover each other and automatically create virtual links between them. In this simple case the three created virtual links mapped directly to the respective physical links between the devices. The link between the two PCs was negotiated to be a secure link with 128-bit AES encryption. However the links between the PCs and the sensors were not encrypted, since the sensor implementation does not support encryption yet.

Several ping tests were conducted between the devices to verify the connectivity between the devices. Figure B.11 shows a screenshot of the output of the ping application on the USB interface of the sensor that started the ping test. As one can see in this screen shot, the addresses used by the ping application is the virtual node addresses (nodeID). In fact, the ping application, does not know, whether the destination is another resource-constrained or a non-constrained device. All that is needed for the application is the nodeID of the destination.

In the table B.1 the average round trip times are shown along with the standard deviation for a sample of 100 pings between a sensor at one end of the network and the other devices in the network. It is worth noticing, that the largest time in the path is the time to reach the first hop and get the reply from it (125ms). This is due to the fact, that the sender is a resource-constrained device and it had a few debug options turned on in order to see what is going on in the test. The next part along

Table B.1: Ping round trip times in millisecond.

Hop Count	1	2	3
Round Trip Time [millisecond]	125	154	216
Standard Deviation	75	92	98

the path added 29ms to the round trip time. This time is less than one fourth of the time for the first part, although the packet in this part needed more processing power since it was encrypted and decrypted. The reason for this is that this part is between two PCs, which of course have more processing power than the sensors. The third part of the path added 62ms to the round trip time. Although this part of the network is very similar to the first part, the round trip time is about half the time of the first part along the path. The main reason for this reduced time is that this second sensor had all debug options turned off in contrast to the first sensor.

The ping tests demonstrated, that it is possible to communicate between two sensors, that belong to different sensor networks, as long as the sensors belong to the same IoT-VN. Furthermore it demonstrated that end-to-end communication between all devices inside an IoT-VN, regardless whether resource-constrained or not, is carried out without any protocol translation.

Figure B.12 illustrates the packet flow of the ping packet as it travels its path from the source of the ping to its destination over the virtual network. The ping packet is prepended with a IoT-VN network header to facilitate routing of the packet in the virtual network. The sending sensor also prepends the 802.15.4 header to enable the packet to travel over the sensor networks. When the first pc receives the packet it strips the 802.15.4 header and sends it to the IoT-VN application to decide on the next hop in the virtual network. Since in this case the next hop is the second PC, the data should be sent encrypted over the link. The PC calculates the encrypted data and adds the necessary security headers and trailers and prepends them with the Ethernet header in order to send the packet over Ethernet. The packet continues its way along its path following the same rules.

The implemented sensor application (including MAC, AODV routing, IoT-VN functionality and ping) has a footprint of 35508 bytes in ROM and 4981 bytes in RAM. This footprint fits even on very resource-constrained devices.

The test results and the small footprint of the implementation demonstrate that it is feasible to realize our approach in including resource-constrained and non-constrained devices in one virtual network. However, the implementation is still at an early stage, mainly demonstrating the concept, its feasibility and possible applications. In order to really allow an in-depth evaluation of the IoT-VN concept, several hurdles need to be overcome. For example, the current solution requires a non-constrained device at the border of the network and does not foresee end-to-end tunneling as used in [11]. At this stage, security mechanisms have not been

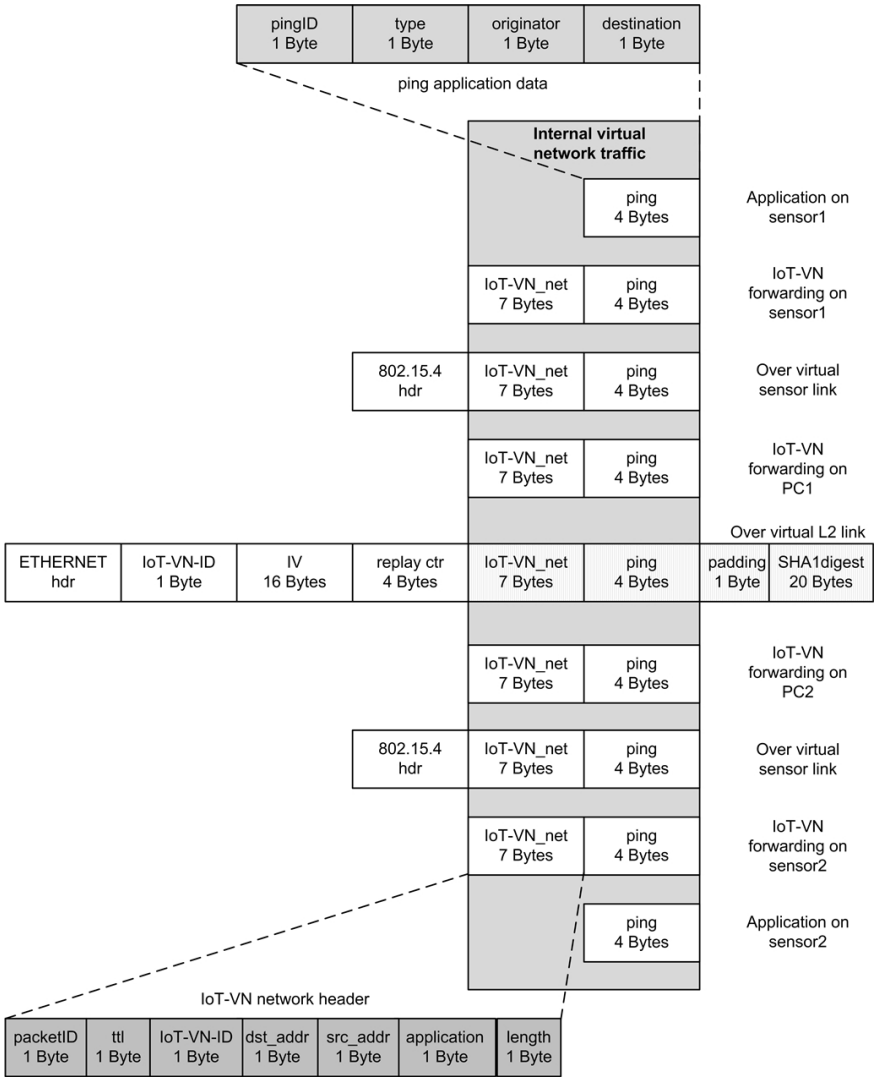


Figure B.12: A ping packet traveling over a virtual network. The ping packet is prepended with a IoT-VN network header to facilitate routing of the packet in the virtual network. When traveling over sensor networks the 802.15.4 header is prepended. In addition to the Ethernet header security headers and trailers are added before sending the packet over Ethernet.

applied yet and will definitely increase the footprint. The addition of security in the near future, based on available research works, will enable us to tackle other related problems such as key distribution, trust establishment and overlay management and would allow more thorough conclusions about the realization of the IoT-VN concept. Another limitation is that the evaluation is based on a very small test bed. This only allows to demonstrate the feasibility, not however to evaluate the performance of the approach in real settings. For performance evaluation a bigger testbed has to be used.

B.7 Conclusions and outlook

In this paper we have introduced our approach to complement existing methods of integrating sensor networks into the Internet. Our approach focuses on the objects that need to cooperate by integrating them into a secured virtual network. Inside this virtual network full end-to-end communication can take place between the networked objects regardless whether they are resource-constrained or not. This is achieved through the use of protocols that take into account the limitations of the most resource-constrained devices. We described how this concept can constitute a valid alternative approach for realizing certain real-life scenarios by providing some several generic use cases. Finally we described our first implementation demonstrating the key concepts of this approach. It proved the feasibility of our approach, but also revealed remaining challenges.

Security is an essential part of our approach. While an acceptable level of security was achieved in the communication between non-constrained nodes in our virtual network, secure communication between resource-constrained devices and between resource-constrained and non-constrained devices remains a challenge. In the future, we plan to implement secure communication between sensors and between sensors and non-constrained devices by using encryption.

The scalability of our approach has not been tested yet. We are planning to test it in a larger-scale, real-life environment by using a wireless sensor testbed, such as the w-iLab.t [19].

Additionally, we plan to investigate to what extend any manual configuration that is still required to create and manage the virtual network can be avoided.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n58885 (SPITFIRE project), from the IBBT ICON project GreenWeCan, and a VLIR PhD scholarship to Isam Ishaq.

References

[1] J. Hoebeke, E. De Poorter, S. Bouckaert, I. Moerman, and P. Demeester. *Managed ecosystems of networked objects*. Wireless Personal Communications, 58(1):125–143, 2011.

[2] N. Kushalnagar, G. Montenegro, and C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. RFC 4919 (Informational), August 2007. Available from: http://www.ietf.org/rfc/rfc4919.txt.

[3] *Routing Over Low power and Lossy networks (roll)*, 2012. [Online; accessed 11 September 2012]. Available from: http://datatracker.ietf.org/wg/roll/.

[4] Z. Shelby. *Constrained RESTful Environments (CoRE) Link Format*. RFC 6690 (Proposed Standard), August 2012. Available from: http://www.ietf.org/rfc/rfc6690.txt.

[5] Z. Shelby, K. Hartke, and C. Bormann. *Constrained Application Protocol (CoAP)*, 2012. [Online; accessed 3 June 2012]. Available from: http://tools.ietf.org/html/draft-ietf-core-coap-08.

[6] O. Garcia-Morchon, S. Keoh, S. Kumar, R. Hummen, and R. Struik. *Security Considerations in the IP-based Internet of Things*, 2012. [Online; accessed 3 June 2012]. Available from: http://tools.ietf.org/html/draft-garcia-core-security-04.

[7] *The VITRO/FP7 project*, 2012. [Online; accessed 3 April 2012]. Available from: http://www.vitro-fp7.eu/.

[8] A. P. Jayasumana, Q. Han, and T. H. Illangasekare. *Virtual sensor networks-A resource efficient approach for concurrent applications*. In Information Technology, 2007. ITNG'07. Fourth International Conference on, pages 111–115. IEEE, 2007.

[9] J. Freeman and D. Passmore. *The virtual lan technology report*. Decisys, Inc., Sterling, VA, 1996.

[10] S. Khanvilkar and A. Khokhar. *Virtual private networks: an overview with performance evaluation*. Communications Magazine, IEEE, 42(10):146–154, 2004.

[11] J. Hoebeke. *Adaptive ad hoc routing and its application in Virtual Private Ad Hoc Networks*. PhD thesis, Ghent University, 2007.

[12] S. Raza, S. Duquennoy, T. Chung, T. Voigt, U. Roedig, et al. *Securing communication in 6LoWPAN with compressed IPsec*. In Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on, pages 1–8. IEEE, 2011.

[13] A. Liu and P. Ning. *TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks*. In Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on, pages 245–256. IEEE, 2008.

[14] *Security of TinySec*, 2012. [Online; accessed 3 September 2012]. Available from: http://www.cl.cam.ac.uk/research/security/sensornets/tinysec/.

[15] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. *The Click modular router*. ACM Transactions on Computer Systems (TOCS), 18(3):263–297, 2000.

[16] E. De Poorter, E. Troubleyn, I. Moerman, and P. Demeester. *IDRA: A flexible system architecture for next generation wireless sensor networks*. Wireless Networks, 17(6):1423–1440, 2011.

[17] *TinyOS*, 2012. [Online; accessed 3 April 2012]. Available from: http://www.tinyos.net/.

[18] C. Perkins, E. Belding-Royer, and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561 (Experimental), July 2003. Available from: http://www.ietf.org/rfc/rfc3561.txt.

[19] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester. *The w-iLab. t testbed*. In Testbeds and Research Infrastructures. Development of Networks and Communities, pages 145–154. Springer, 2011.