FREIE UNIVERSITÄT BERLIN

Department of Mathematics and Computer Science
Institute of Computer Science

Doctoral Dissertation

Information-centric Networking for the Constrained Internet of Things

Dissertation zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.) am
Fachbereich Mathematik und Informatik der Freien Universität Berlin

vorgelegt von

M.Sc. Cenk Gündoğan
cenk.guendogan@fu-berlin.de
Berlin 2022

Erstgutachter
Prof. Dr. Matthias Wählisch
Freie Universität Berlin

Zweitgutachter
Prof. Dr. Thomas C. Schmidt
Hamburg University of Applied Sciences

Drittgutachter
David R. Oran
Network Systems Research & Design

Tag der Disputation
19.07.2022

Abstract

Information-Centric Networking (ICN) promises an enhanced reliability for content retrievals in the Internet of Things (IoT), while reducing link stress and network-related energy expenditure. Wireless, low-power regimes, however, pose challenging environments to present-day ICN IoT deployments, which provides grounds for rethinking how information-centric principles integrate into the resource-constrained IoT. The principal aspiration of this thesis is to revisit the constrained ICN deployment by putting emphasis on wireless and harsh deployments with very low resource capacities to achieve a reliable and secure data delivery that scales with the number of network participants.

Part I of this manuscript develops a protocol suite for the low-power IoT to reduce memory demands, improve the utilization of wireless links, and lower the power consumption for information-centric content retrievals. A new convergence layer follows the design elements of IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN), and adapts ICN packets to the small-sized Maximum Transmission Units (MTUs) of low-power links by providing a header compression scheme, link fragmentation, and protocol framing similar to EtherTypes. A lightweight Quality of Service (QoS) scheme on the network layer complements this protocol suite. It enables a coordinated resource management to reduce network latency, and to prevent queue starvation for prioritized traffic flows. Since device mobility and intermittent connectivity are prevalent in these regimes, a new publish–subscribe system bolsters the information-centric IoT against network disruptions, and improves routing agility on connectivity loss.

Part II examines the Internet perspective of native ICN IoT networks, and then describes the construction of a data-centric Web of Things (WoT) to lead insights and techniques emerging from ICN research into a promising, realistic deployment trail for the growing IoT. This deployment option is based on standard protocol elements of the Constrained Application Protocol (CoAP), and reflects the three information-centric principles (*i*) stateful forwarding, (*ii*) hop-wise caching, and (*iii*) content object security. Real protocol implementations and testbed assessments on actual IoT hardware show that the data-centric WoT adheres to performance expectations of pure ICN deployments, while retaining full compatibility with Internet services.

Zusammenfassung

Information-Centric Networking (ICN) verspricht eine zuverlässige Datenübertragung mit reduziertem Energieaufwand für das Internet of Things (IoT). Leistungsarme und verlustbehaftete Drahtlosnetzwerke stellen jedoch eine Herausforderung für die gewöhnliche Nutzung informationszentrischer Technologien dar. Das Hauptziel dieser Arbeit ist es, die herkömmliche Integration von ICN im ressourcenbeschränkten und drahtlosen IoT zu überdenken, damit eine zuverlässige, sichere und skalierbare Datenübertragung gewährleistet werden kann.

Teil I dieses Manuskripts entwickelt eine Protokollsuite für das informationszentrische IoT. Im Fokus steht die Reduzierung des Speicherbedarfs, die Entlastung der verlustbehafteten Drahtlosverbindungen und die Senkung des netzwerkorientierten Stromverbrauchs für batteriebetriebene Geräte. Eine neue Konvergenzschicht folgt den Entwurfselementen von IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) und passt ICN-Pakete an die beschränkte maximale Übertragungseinheit von energiearmen Funktechnologien an, indem sie ein Kompressionsschema und eine Paketfragmentierung bereitstellt. Ein leichtgewichtiger Quality of Service (QoS) Ansatz auf der Netzwerkschicht ergänzt diese Protokollsuite. Dieser ermöglicht eine koordinierte Ressourcenverwaltung zur Verringerung der Netzwerklatenz und zur Verhinderung des Aushungerns von priorisierten Verkehrsströmen. Da Geräte in herausfordernden Drahtlosumgebungen meist mobil sind und die Konnektivität auch durch Interferenzen recht diskontinuierlich sein kann, unterstützt ein neues Publish-Subscribe-System das informationszentrische IoT gegen Netzwerkunterbrechungen und verbessert die Agilität des Routings bei Konnektivitätsverlust.

Teil II untersucht die Internet-Perspektive herkömmlicher ICN IoT Netzwerke und beschreibt dann die Konstruktion eines datenzentrischen Web of Things (WoT). Dieses Konstrukt ermöglicht die Überführung von Erkenntnissen und Techniken aus der ICN-Forschung in einen vielversprechenden, realistischen Einsatzpfad für das heutige, wachsende IoT. Diese neue Einsatzoption basiert auf den Standardprotokollelementen des Constrained Application Protocol (CoAP) und spiegelt die drei informationszentrischen Prinzipien wider: (*i*) zustandsbehaftetes Forwarding, (*ii*) hopweise Caching und (*iii*) inhaltsbasierte Sicherheit. Reale Protokollimplementierungen und Testbed-Bewertungen auf tatsächlicher IoT-Hardware zeigen, dass das datenzentrische WoT die Leistungserwartungen reiner ICN-Implementierungen erfüllt und dabei die volle Kompatibilität mit klassischen Protokollen und Diensten rund um das Internet beibehält.

Selbstständigkeitserklärung

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertig habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Berlin, den 19. Juli 2022

Cenk Gündoğan

Acknowledgments

The road towards a PhD often appears lonely and solitary, but, in retrospect, this endeavor would not exist without the professional and emotional support of the many people that crossed my path.

I wish to express my deepest gratitude to my supervisors Thomas Schmidt and Matthias Wählisch for putting me on this marvelous journey. Their counsel was invaluable, their support was continuous, and their patience during my academic studies was endless. Our discussions and collaborations encouraged me in all the venture and helped me to grow not only professionally, but also on a personal level.

I also would like to pay special regards to David R. Oran, who wholeheartedly reviewed this manuscript. His stellar work in the ICN community and his inspiring comments were always a learning opportunity for broadening my knowledge on networking aspects.

I also would like to recognize the monumental assistance of the iNET working group—my colleagues and dear friends. Peter Kietzmann was the first to welcome me in Hamburg and I will always be grateful for his great comradeship throughout all those years and the years to come. It was a pleasure to share an office with him and Michel Rottleuthner, who is knowledgeable about so many software and hardware related topics. Thank you for being a great sparring partner in all our discussions! José Alamos was passionate about everything I conversed with him, and consistently fueled my enthusiasm on many professional and personal topics, thank you, weón! Special thanks also to Leandro Lanzieri, who was always there when I needed another opinion or a change in perspective. I will not forget our various off-work activities and will remember you as my personal mixologist, boludo! Can a person be practical and radical at the same time? Kevin Weiss can. Thank you, buddy, for sharing your pragmatic and thought-provoking views on everything we discussed. I also express my gratitude to Raphael Hiesgen for being a reliable colleague and an always prepared co-admin. His love for detail when reviewing my presentation slides most certainly had an impact. All the kudos to you, dear friend. Sebastian Meiling quickly became a well-respected expert in all matters around RIOT, and I am grateful for all his input in subjects related to IoT, server maintenance, and teaching. I would also like to thank Lena Boeckmann, Jasper Eumann, Timo Häckel, Philipp Meyer, Jakob Otto, Marcel Röthke, and Colin Sames for the exciting moments during lunchtime and during our iNET seminars. Self-discipline and stress-management are character traits that I picked up from the exceptional stoic Bobo. Thank you for always being the first in the office and being the last to lock the doors. I will cherish your wisdom, chum!

I am also indebted to Emmanuel Baccelli, Oliver Hahm, Martine Lenders, and Hauke Petersen who I know from a time when I was still studying at the Freie Universität Berlin. Their expertise in RIOT is endless and their guidance on technical aspects actually helped me to embrace the fields of embedded programming and networking constrained things. Pouyan Fotouhi Tehrani has an upright personality and questions everything you throw at him. I really enjoyed our philosophical and political conversations. A big shout-out also to Marcin Nawrocki, who was always a great host when I visited Berlin and allowed me to sit in his office. I met Michael Frey and Felix Shzu-Juraschek during my undergraduate studies, and it was refreshing to see you both appear again as colleagues on the same research project.

I would further like to extend my sincere gratitude to Christian Amsüss for his insightful comments and suggestions on several papers. He taught me a lot about the protocol landscape around CoAP, and it was a pleasure to explore new concepts with him. Jakob Pfender generously shared his experiences on content caching in the IoT, and I am thankful for our collaboration. Special thanks to Claudio Marxer, Christopher Scherb, and Christian Tschudin for their work on CCN-lite. A substantial part of my experimentation effort would not exist without this tool. I also wish to thank all the members of the ICN research group for fruitful in-person and online discussions, especially Alex Afanasyev, Dirk Kutscher, Marc Mosko, and Lixia Zhang.

Finally, I wish to offer my sincere thanks to my parents, who taught me to reach for the stars and stand up when I fall, and Nataliia, for accompanying me on this journey with all her love and patience.

The work in this thesis was supported in part by the German Federal Ministry for Education and Research (BMBF) within the projects *I3 – Information Centric Networking for the Industrial Internet*, *RAPstore – RIOT App Store*, *PIVOT – Privacy-Integrated design and Validation in the constrained IoT*, and the Hamburg *ahoi.digital* initiative with *SANE.*

Bibliographical Notes

This dissertation is based on the following list of research papers. The author of this thesis derived and formulated the research goals, designed the majority of the solution space, implemented and conducted significant parts of the experiments, performed the data curation, and led the writing and visualization of the paper.

Part 1

- Chapter 3 is based on

  C. Gündogan, P. Kietzmann, M. S. Lenders, H. Petersen, M. Frey, T. C. Schmidt, F. Shzu-Juraschek, and M. Wählisch, "The Impact of Networking Protocols on Massive M2M Communication in the Industrial IoT," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 18, no. 4, pp. 4814–4828, (Piscataway, NJ, USA), IEEE, December 2021. `https://doi.org/10.1109/TNSM.2021.3089549`.

- Chapter 4 is based on

  C. Gündogan, P. Kietzmann, T. C. Schmidt, and M. Wählisch, "Designing a LoWPAN convergence layer for the Information Centric Internet of Things," *Computer Communications*, vol. 164, pp. 114–123, Elsevier, December 2020. `https://doi.org/10.1016/j.comcom.2020.10.002`.

- Chapter 5 is based on

  C. Gündogan, J. Pfender, P. Kietzmann, T. C. Schmidt, and M. Wählisch, "On the Impact of QoS Management in an Information-centric Internet of Things," *Computer Communications*, vol. 154, pp. 160–172, Elsevier, March 2020. `https://doi.org/10.1016/j.comcom.2020.02.046`.

- Chapter 6 is based on

  C. Gündogan, P. Kietzmann, T. C. Schmidt, and M. Wählisch, "A Mobility-compliant Publish Subscribe System for an Information Centric Internet of Things," *Computer Networks*, vol. 203, Elsevier, February 2022. `https://doi.org/10.1016/j.comnet.2021.108656`.

Part 2

- Chapter 8 is based on

  C. Gündogan, C. Amsüss, T. C. Schmidt, and M. Wählisch, "Reliable Firmware Updates for the Information-Centric Internet of Things," in *Proc. of 8th ACM Conference on Information-Centric Networking (ICN)*, (New York), pp. 59–70, ACM, September 2021. `https://doi.org/10.1145/3460417.3482974`.

- Chapter 9 is based on

  C. Gündogan, C. Amsüss, T. C. Schmidt, and M. Wählisch, "Content Object Security in the Internet of Things: Challenges, Prospects, and Emerging Solutions," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 19, no. 1, pp. 538–553, (Piscataway, NJ, USA), IEEE, March 2022. `https://doi.org/10.1109/TNSM.2021.3099902`.

- Chapter 10 is based on

  C. Gündogan, C. Amsüss, T. C. Schmidt, and M. Wählisch, "Toward a RESTful Information-Centric Web of Things: A Deeper Look at Data Orientation in CoAP," in *Proc. of 7th ACM Conference on Information-Centric Networking (ICN)*, (New York), pp. 77–88, ACM, September 2020. `https://doi.org/10.1145/3405656.3418718`.

- Chapter 11 is based on an extended version of

  C. Gündogan, C. Amsüss, T. C. Schmidt, and M. Wählisch, "Group Communication with OSCORE: RESTful Multiparty Access to a Data-Centric Web of Things," in *Proc. of the 46th IEEE Conference on Local Computer Networks (LCN)*, (Piscataway, NJ, USA), IEEE Press, October 2021. `https://doi.org/10.1109/LCN52139.2021.9525000`.

Contents

Acronyms

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks.

ACK Acknowledgment.

CCNx Content-Centric Networking.

CDN Content Delivery Network.

CoAP Constrained Application Protocol.

CS Content Store.

CSMA/CA Carrier-Sense Multiple Access with Collision Avoidance.

DNS Domain Name System.

DODAG Destination-Oriented Directed Acyclic Graph.

DONA Data-Oriented Network Architecture.

DoS Denial of Service.

DTLS Datagram Transport Layer Security.

DTN Delay-Tolerant Networking.

FIB Forwarding Information Base.

HMAC Keyed-hash Message Authentication Code.

HoPP HoP-and-Pull.

HTTP Hypertext Transfer Protocol.

ICN Information-Centric Networking.

ICNLoWPAN ICN over Low-Power Wireless Personal Area Networks.

IETF Internet Engineering Task Force.

IIoT Industrial IoT.

IoT Internet of Things.

LLN Low-power and Lossy Network.

LoWPAN Low-Power Wireless Personal Area Network.

LwM2M Lightweight Machine to Machine.

MAC Medium Access Control.

MQTT Message Queuing Telemetry Transport.

MQTT-SN Message Queuing Telemetry Transport for Sensor Networks.

MTU Maximum Transmission Unit.

NDN Named-Data Networking.

NDO Named-Data Object.

NetInf Network of Information.

OMA Open Mobile Alliance.

OneDM One Data Model.

OSCORE Object Security for Constrained RESTful Environments.

PIT Pending Interest Table.

PSIRP Publish Subscribe Internet Routing Paradigm.

PURSUIT Publish Subscribe Internet Technology.

QoS Quality of Service.

REST Representational State Transfer.

RPL Routing Protocol for Low-Power and Lossy Networks.

SDF Semantic Definition Format.

SDN Software Defined Networking.

SUIT Software Updates for Internet of Things.

TCP Transmission Control Protocol.

TD Thing Description.

TLV Time-Length-Value.

TRIAD Translating Relaying Internet Architecture integrating Active Directories.

UART Universal Asynchronous Receiver-Transmitter.

UDP User Datagram Protocol.

URI Uniform Resource Identifier.

W3C World Wide Web Consortium.

WoT Web of Things.

WSN Wireless Sensor Network.

Chapter 1

Introduction

1.1 Inter-networking Resource-constrained Things

The adoption of numerous Internet of Things (IoT) use cases is on the rise and scenarios like predictive maintenance, smart metering, or asset tracking raise requirements for network performance, energy efficiency, scalability, and security. Sensors and actuators are the principal constituents of the IoT and from today's perspective, these nodes often comprise class 2 devices [1]. They have limited computational power and show scarce memory capacities of up to ≈ 50 KiB main memory. While sensors take on the role of content producers by measuring the physical environment, actuators move or control a system by either receiving external instructions, or by autonomous decision-making. In many scenarios, these *things* have no permanent power supply, because they are either mobile in every-day use, or they are remotely deployed and hardly maintainable. Although an extensive radio use greatly dominates the energy expenditure, and thus the operational lifespan of battery-driven devices, they interconnect wirelessly to form Low-power and Lossy Networks (LLNs) [1]. These regimes are characterized by low bandwidths and high latencies, while Maximum Transmission Units (MTUs) of low-power links are generally small-sized, *e.g.,* 127 bytes for IEEE 802.15.4 [2]. Uncontrollable radio interference, extremely prolonged media utilization, and difficulties in the Medium Access Control (MAC) sublayer, *e.g.,* hidden terminal problem, are a few reasons of many that prevent successful packet transmissions in LLNs. Most MAC technologies provide a positive Acknowledgment (ACK) mechanism with retransmissions to cope with packet loss on a single link.

The Prevalent IoT Network Stack. Limited energy budgets, low bandwidth, high latency, and deficient computational power challenge network infrastructures up to a point, where general purpose Internet technologies become infeasible for the IoT. This fostered early vendor-specific silo solutions that are deeply tailored to single tasks and hardly interoperable. One and a half decades ago, the *6lowpan* working group of the Internet Engineering Task Force (IETF) initiated work on bridging the interoperability gap between the Internet and various IoT deployments by identifying necessary features to extend the Internet into the low-power domain. An outcome of these efforts is the IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [3] convergence layer to adapt the Internet protocol (IPv6) to link-layer technologies with small energy footprints. It harbors three core protocol extensions. (*i*) Stateless and stateful header compression schemes to reduce transmission overhead, increase the overall goodput, and decrease latency. (*ii*) An application-agnostic link fragmentation to enable packet deliveries for larger payloads on link-layers with small-sized MTUs. (*iii*) A dispatching logic for link technologies that lack support for proper frame encapsulation formats (*c.f.,* EtherTypes in Ethernet).

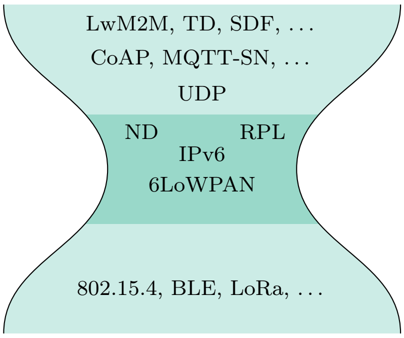


Figure 1.1: Building blocks and protocols to form an IoT network stack.

Since then, several IETF working groups emerged to address problems on different layers of the IoT network stack as illustrated in Figure 1.1. The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) builds multi-hop routing topologies that optimize for the prevalent *convergecast* pattern—*i.e.,* multiple devices connect to a central gateway node—for the cost of installing less efficient routes for a device-to-device communication within the IoT stub network. The Constrained Application Protocol (CoAP) [4] contributes Representational State Transfer (REST) operations to IoT applications. It also provides a separate end-to-end reliability layer based on positive ACKs with retransmissions for deployments over unreliable, best-effort transports, such as UDP. Latest developments extend CoAP with block-wise transfer features and protective measures on the content object level to enable an end-to-end security across untrusted gateways. Message Queuing Telemetry Transport (MQTT) [5], and more importantly its constrained adaptation Message Queuing Telemetry Transport for Sensor Networks (MQTT-SN) [6], is an OASIS standard for IoT communication. It uses a lightweight publish-subscribe layer, has a small memory footprint, and requires minimal network bandwidths. Likewise, MQTT-SN provides reliability using end-to-end retransmissions over a UDP transport. Due to its auspicious characteristics, it enjoys a wide deployment at various renowned cloud services.

Names in the IoT. CoAP operates similarly to Hypertext Transfer Protocol (HTTP) and uses Uniform Resource Identifiers (URIs) to identify services on CoAP endpoints. To extend the range of capabilities of the IoT network stack, more and more application protocols leverage the RESTful operations that CoAP is providing. Open Mobile Alliance (OMA) builds Lightweight Machine to Machine (LwM2M) on CoAP for managing IoT nodes and allowing for a vendor independent co-existence of IoT ecosystems. At the same time, World Wide Web Consortium (W3C) develops a descriptive format to model *things* and their interactions using service URIs. The Thing Description (TD) [7] is an integral part of the W3C initiative to build a Web of Things (WoT) that shall de-fragment the IoT by applying existing, standardized web technologies and other technological building blocks. One Data Model (OneDM) is conducting an adjacent undertaking with Semantic Definition Format (SDF) [8] by harmonizing the landscape of semantic IoT models to foster an interoperable environment for devices and services across vendors and standards organizations. With the evolution of these web technologies to enable a semantic WoT, names become an integral part of the application design. The IETF CoRE working group further defines a central Resource Directory [9] to assist with the discovery of these names in networks of sleepy devices.

Alternative Approaches to IoT Networking. The IoT network stack as illustrated in Figure 1.1 is by design compatible with the Internet, and builds in essence on its end-to-end transport principle. Research communities also examined alternative solutions—most notably Delay-Tolerant Networking (DTN) [10] and Information-Centric Networking (ICN) [11, 12]. While the former utilizes convergence layer adaptations to prevalent Internet transports, the latter replaces the entire network layer to enable enhancements for content retrievals. Due to its potential, this thesis focuses on the utilization of ICN technologies for the constrained IoT.

1.2 Information-Centric Networking

Rapid advancements on the Internet call for a scalable and efficient distribution of content items, such as web pages, documents, and videos. An accelerating growth of the IoT and its amassed sensory data further bring forward new distributed Denial of Service (DoS) attacks against the Internet infrastructure. Proprietary Content Delivery Network (CDN) overlays became attractive to cope with the demanding requirements of data distribution and to mitigate the effects of DoS attacks. They employ caching techniques to pull popular content closer to consumers, and thus improve on the responsiveness at reduced networking cost. Nevertheless, overlay solutions can quickly reach the limitations of the underlying network technology, which raises the question for an architectural redesign of the Internet to efficiently address the scalability and security issues directly on the network.

One major outcome of the global future Internet initiatives is the ICN paradigm. A variety of ICN flavors have been created during more than a decade of research. These variants essentially have three principles in common [13]: (*i*) *Decoupling of named content from hosts*, (*ii*) *universal caching*, and (*iii*) *content object security.* The approaches separate content access from physical infrastructure, allow for unhindered content replication and validation, and thereby reduce infrastructure dependency of the content-aware network layer.

Translating Relaying Internet Architecture integrating Active Directories (TRIAD) [14] pioneered the approach to information-centric content retrievals by introducing name-based routing. Routers direct requests towards content servers and caches using dedicated content names instead of endpoint addresses. Data-Oriented Network Architecture (DONA) [15] extends TRIAD with a stateful forwarding mechanism and improves on the data persistence and authenticity with flat, self-certifying names. Publish Subscribe Internet Routing Paradigm (PSIRP) [16] and its successor Publish Subscribe Internet Technology (PURSUIT) [17] present a clean-slate approach to replace IP networking with a publish-subscribe communication model. They em-

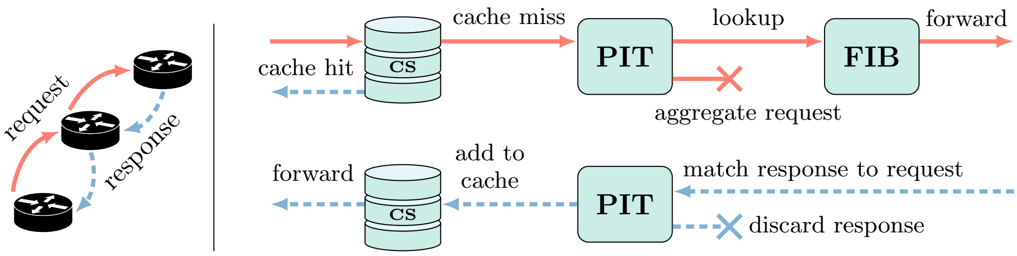


Figure 1.2: Stateful forwarding plane of CCNx and NDN: Aggregable Interests build a reverse path and are served from content caches.

ploy a rendezvous point, which acts as a resolver for data requests. It calculates a return path for the data and instructs the content to move from a publisher to a subscriber using source routing techniques. Network of Information (NetInf) [18] aims for great flexibility by combining two models for retrieving named content. First, it supports a name resolution similar to PURSUIT to retrieve a network locator of the desired content. Second, NetInf offers a name-based routing as employed by TRIAD and DONA.

Prominent ICN architectures. Content-Centric Networking (CCNx) [19, 20] and Named-Data Networking (NDN) [21, 22] enjoy the greatest popularity among the different ICN technologies. They couple the name-based routing from TRIAD with the stateful forwarding from DONA. NDN has an increasing community of mainly academic nature and several open source projects evolved around its concept. Prior work at PARC on CCNx has been transferred to Cisco and was open sourced as the Hybrid-ICN architecture [23]. In contrast to the stateless packet processing of the Internet, CCNx and NDN utilize a stateful, name-based forwarding fabric to achieve a decoupling of content objects from their origins. The network layer uses the two distinct message primitives *Interest* and *Data* to implement a request-response data retrieval model, where each message type is treated differently by the forwarding state machine.

Figure 1.2 illustrates the forwarding logic and involved data structures. Incoming Interests first trigger a cache lookup at the Content Store (CS) on every hop. On a cache hit, a response with the requested content is returned to the previous hop. If there is a cache miss, then the Pending Interest Table (PIT) is queried for a possibly existing request state with the same content name from a previous request attempt. On a positive lookup, the new request is aggregated, *i.e.,* the incoming interface is added to the existing request state and the forwarding terminates on that particular hop. If there is no existing PIT entry for a particular content, then the request traverses a node for the first time. A PIT entry is created to record the content name and the incoming interface. In accordance with the Forwarding Information Base (FIB), the request is forwarded towards one or several outgoing interfaces. Returning responses match against the PIT to identify pending requests, and if no requests are pending, then the response is discarded. Otherwise, the response is cached in the CS and forwarded towards the recorded interfaces in the PIT.

Basic IoT Integration. The IoT, which is mainly composed of constrained nodes at the network edges, is a potential beneficiary of the directions undertaken by CCNx and NDN as they reduce the burden of maintaining dedicated server infrastructure, end-to-end communication channels, and DoS mitigation. In addition, seamless content replication and caching open the realm to multilateral support of energy preservation and improved transport resilience. Realistic IoT deployments of the ICN layer, however, demand additional functions such as service differentiation and publish-subscribe. Foremost, ICN needs to show convincing evidence for network performance superior to traditional approaches.

Two modes of deployment are commonly considered for an information-centric IoT system. First, an ICN is configured as an overlay of the existing IP infrastructure using a Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) encapsulation. Alternatively, ICN takes the role of the network layer and replaces IP. The encapsulation mode is generally discouraged for the IoT, because of inflated packets, its memory overhead, and software complexity of hosting two separate network stacks. The native deployment of ICN on the network layer is thus the primarily viable approach for challenged IoT networks. It allows for lean network stacks of reduced complexity and more thoroughly realizes an information-centric IoT system.

1.3 Research Questions

The principal aspiration of this thesis is to revisit the constrained ICN deployment by putting emphasis on wireless and harsh deployments with very low resource capacities to achieve a reliable and secure data delivery that scales with the number of network participants. The following research questions and challenges will be addressed to converge towards this overarching goal.

1.3.1 Information-Centric Networking for the Internet of Things

Protocol Behaviors in Harsh Environments. IoT communication involves the voluminous retrieval of sensor readings from a large group of IoT devices. The wireless IoT domain is lossy, and error probabilities accumulate on every hop while messages are forwarded. Prominent application protocols, such as CoAP and MQTT-SN, as well as the network technologies CCNx and NDN implement corrective actions to ensure network operability in these regimes. Although these protocols are either specifically designed for IoT communications (CoAP, MQTT-SN), or show potentials that may promote their applicability in the IoT (CCNx, NDN), they nevertheless perform very differently under harsh conditions giving consideration to their significantly different mode of transport: end-to-end versus hop-by-hop. This leads to the following research questions, which are further discussed in Chapter 3.

Research Questions

- How to quantify the efficacy and utility of candidate protocols for the IoT?
- Which protocol variant stands out against rough deployment conditions?

Link-layer Convergence. ICN deployments are designed for efficient transfers of content, but transmissions are challenged by low-power link-layers. The IP-world reacted with the 6LoWPAN convergence layer that adapts IPv6 to these link limitations. This layer provides a link fragmentation scheme, stateless and stateful header compression, and frame encapsulation formats akin to EtherTypes in Ethernet. ICN protocol variants can benefit from similar practices, and may provide additional advantages, *e.g.,* due to its stateful forwarding fabric. Chapter 4 focuses on these challenges and approaches the following research questions.

Research Questions

- Can we build a superior convergence layer for ICN on low-power link-layers?
- Can we enable coexistence between an ICN convergence layer and 6LoWPAN?
- Can we determine and leverage intrinsic protocol characteristics to develop a link adaptation that improves on the IP counterpart?

1.3.2 Information-Centric Networking in Mobile Low-Power Regimes

Quality of Service. IoT networks are often under-provisioned to decrease device complexity, unit price, and energy consumption. Consequently, high peak loads as generated, *e.g.,* by firmware roll-outs are demanding for these networks and challenge their functional operation. Low bandwidth and limited buffer spaces can create destructive situations, in which packet loss escalates due to resource saturation. The IP-world has established solutions to address service differentiation and resource reservations, whereas this topic is mainly unexplored in information-centric systems. It is especially disturbing for latter architectures, since stateful forwarding requires an additional state management, and simply replicating IP-solutions to ICN is considered to yield insufficient results [24]. The following intriguing research questions result from this problem statement and are further addressed in Chapter 5.

Research Questions

- Can we identify resource dimensions in an ICN system?
- Can we define an efficient and scalable service differentiation for the IoT?
- Can we construct a lightweight resource coordination for the IoT based on information-centric principles?

Producer-mobility and Delay-tolerance. In industrial IoT settings, *e.g.,* oil rigs and warehouse facilities, the successful and uninterrupted aggregation of sensor data from hand-held measuring devices is necessary to meet mission-critical requirements and regulatory compliance. For dealing with device mobility and disrupted networks, a loose coupling between nodes is often desired. ICN decouples content provisioning from data producers in space, which makes it a promising candidate. Further decoupling in time and synchronization contributes to network resilience and is attainable by a publish-subscribe layer. This implies the following research questions, which Chapter 6 inspects in more detail:

Research Questions

- Can we leverage ICN to build a robust publish-subscribe system?
- Can we reduce forwarding state requirements to operate in the low-power IoT?
- Can we achieve a high reactivity in connected networks and sufficient delay-tolerance while partitioned?

1.3.3 Information-Centric Principles in the Web of Things

CoAP is the IETF solution to retrieve content in constrained networks and in conjunction with W3C web technologies, deployments form a host-centric, RESTful WoT, which benefits from a wide academic and industrial acceptance. While promising, the information-centric IoT is undeniably less prevalent and misses out on similar deployment experiences that might provide opportunities for protocol evolution. The information-centric principles (*i*) name-based, stateful forwarding, (*ii*) in-network caching, and (*iii*) content object security are advantageous for the IoT, but a native ICN deployment is inherently incompatible with Internet services, not only on the protocol level, but more significantly on the application design. This leads to a fragmented deployment landscape, where application-level gateways are necessary for a basic interconnectivity. Alternatively, overlay solutions can bring an ICN-style communication to IP-based IoT networks, but this generally produces convoluted designs with unacceptable resource overhead. The following questions explore the interoperability problem of efficient content retrievals with prevalent IP-based IoT infrastructure and motivate the investigation of different paths to integrate the information-centric principles into the WoT.

Motivating Reliable Content Distributions in the WoT. General purpose devices require timely software updates and increasing security demands make similar practices equally relevant to the IoT. A secure and reliable firmware update propagation in multi-hop, low-power regimes is, however, challenging. Also for the Internet, software updates are resource-demanding and appear as peak loads. The Software Updates for Internet of Things (SUIT) working group of the IETF defines a firmware update architecture, although not the actual discovery or transport of firmware images. At the same time, IoT networks are typically designed to serve sensor data of a few bytes, but these images are two to three orders of magnitude larger. A firmware roll-out, thus, quickly hits bandwidth limitations and occupies device and network resources for extended periods, which can impact network serviceability and nodal lifetimes. Information-centric content retrievals pave the way for a reliable and efficient distribution of massive content. Chapter 8 presents a use case that elaborates on an information-centric firmware-rollout in low-power regimes and aims for addressing the following questions. The objective of this use case is to motivate a continuing effort to enable a data-centric access to (large) content objects in an interoperable manner for the WoT.

Research Questions

- Can we leverage information-centric properties to perform secure and reliable firmware-rollouts at large-scale for the IoT?
- Can we securely transmit voluminous data, while keeping the overhead of protective measures low?

Security Model for the IoT. One significant aspect that advances data orientation further into perspective relates to the IoT security model. Transport security is the predetermined approach to protect data streams on the Internet. Despite its endpoint based session management complexity, the inability to sustain end-to-end protection beyond protocol translating gateways, and the inconveniences it faces in mobile scenarios and group communication, it finds an increasing acceptance in constrained environments with Datagram Transport Layer Security (DTLS) [25]. ICN was first to introduce content object security on the network layer for the sake of ubiquitous caching. Recently, the IETF Core working group released Object Security for Constrained RESTful Environments (OSCORE) [26], which extends the IoT ecosystem around CoAP with content object security. This new approach casts doubts on using the pervasive transport security model of the Internet for the IoT and raises the following question, which is conceptually addressed by Chapter 9.

Research Questions

- Is OSCORE a superior alternative to DTLS for secure networking in the IoT?

?

Information-centric Principles for the WoT. Stateful forwarding and in-network caching are integral properties of prominent ICN architectures. These constituents make ICN appealing to the constrained IoT as infrastructural burdens and common DoS threats, which have established in the current Internet, stand in the way of a lean and efficient inter-networking for embedded devices. This questions the widely adopted practice to IoT communication. The following research challenges are addressed in more detail in Chapter 10.

Research Questions

- Can we build a RESTful information-centric WoT with CoAP?
- Will it improve robustness for content retrievals compared to a host-centric WoT?

?

Multiparty Communication for the Information-centric WoT. Many IoT use cases require a reliable and efficient multiparty transmission to either obtain sensor readings from multiple sources, or to propagate common instructions to multiple sinks. An efficient solution to group communication can significantly decrease the communication overhead and energy consumption, as much as inconsiderate approaches can increase signaling overhead and energy demands. Especially in wireless regimes, scalable network solutions are required, since superfluous transmissions unnecessarily occupy shared broadcast media. The traditional approach to multiparty communication on the Internet is to use IP multicast as a connectionless, best-effort service. This is not easily transferable to the IoT domain, because typical IoT link-layers lack efficient multicast mappings, and default to broadcasting instead. Further, lossy networks have higher packet loss rates, and corrective actions for IP multicast involve a serious signaling overhead. In ICN architectures, the loose coupling of content and data producers facilitates multi-source and multi-destination traffic flows. Chapter 11 inspects the following open research challenge to an efficient group communication more thoroughly.

Research Questions

- Can we enable a reliable and secure multiparty communication built on information-centric properties for the RESTful WoT?

?

1.4 Methodology

IoT protocols demand careful design considerations, since limited link and device resources result in an unforgiving environment. A few subpar decisions can already impact the network performance and battery lifetimes. Theoretic models and simulations provide insights on protocol effects, but the number of variables increases significantly for realistic predictions and observations due to resources that correlate locally as well as across several neighboring nodes. A high network utilization and node stress—which are both quickly attained in large-scale deployments of moderate traffic—lead to spurious events that disturb expected protocol behaviors. To explore the event space and assess performances under realistic conditions, experimentally-driven research methods are unavoidable and can validate appropriate input parameters for alternative methods.

Theoretical Estimations. The validity of selected hypotheses throughout this manuscript were tested theoretically prior to running extensive, practical assessments. The theoretical evaluations were approached in one of two ways: (*i*) A formal description of the problem was designed that arithmetically estimates key properties. Goodput calculations, compression ratios, protocol overhead, and collision probabilities were assessed using this method. (*ii*) Qualitative evaluations with conceptual analytical models allowed for the comparison of feature sets and protocol effects in complex scenarios that are difficult to realize practically.

Experimental Evaluation. The contributions in Part I and II evaluate all hypotheses on real IoT nodes managed by the FIT IoT-LAB testbed [27] to reflect common IoT properties. The testbed consists of several hundred class 2 [1] devices equipped with an ARM Cortex-M3 MCU, 64 kB of RAM, 512 kB of ROM, and an IEEE 802.15.4 radio. The radio module provides basic MAC layer functions implemented in hardware, such as ACK handling, retransmissions, and Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA). The tooling around the testbed allows for various ways to interact with single nodes. In all subsequently presented evaluations, the experimental data was gathered from the embedded devices either via the Universal Asynchronous Receiver-Transmitter (UART) peripheral, or through an external, wireless packet sniffer.

Routing Topologies and Performance Metrics. The quantitative protocol analyses throughout this manuscript illuminate protocol elements under varying traffic loads and routing topologies. Delicate multi-hop deployments of differing sizes are either statically constructed, or dynamically built by a routing protocol. All topologies, however, form a Destination-Oriented Directed Acyclic Graph (DODAG) to optimize for the prominent convergecast traffic pattern, *i.e.,* sensor readings converge from many IoT sources towards a single sink, which often represents a gateway with uplink connectivity.

Selected performance metrics recur in the majority of evaluations. *Time to content arrival* gauges the end-to-end latencies for content replication, while *success rates* measure the ability of protocols to recover message loss in high network stress situations. The *goodput* indicates the rate of receiving actual application payload without the variable header overhead for distinct protocol deployments.

Open-source Software Platform. The decision for a software platform that can cope with limited device resources is crucial. To promote maintainability and sustainability of software components, functional solutions were designed for and implemented on existing code bases instead of reinventing elaborate abstractions from scratch. As such, all experimental protocol assessments in this document use the open-source IoT operating system RIOT [28]. This ensures aligned protocol comparisons where all deployments interface to common network stack components and hardware peripherals, thereby reducing the number of variable timings. Moreover, the versatile open-source community proves invaluable with its support on technical implementation issues.

The default RIOT network stack—Generic (GNRC)—follows a cleanly layered architecture [29]. It provides IPv6 connectivity and implements the 6LoWPAN convergence layer. RPL can be enabled to construct a routing system in a meshed topology across multiple hops. The application protocols CoAP and MQTT-SN build on the UDP socket layer of GNRC. While the integrated `tinyDTLS` library can optionally protect the UDP transport, `libOSCORE`[1] can be leveraged to secure CoAP on the content object level. The integrated `CCN-lite` package brings a lightweight NDN forwarder to RIOT with a simple content store implementation that utilizes the main memory.

1.5 Document Outline

This manuscript provides contributions to address the research questions of Section 1.3. The individual contributions as illustrated in Figure 1.3 are organized into two main parts.

Part I assesses the eligibility and improves the protocol integration of the information-centric variant NDN for networks that are challenged by constrained hardware resources and low-power radio aspects. Chapter 3 compares NDN with CoAP and MQTT-SN—two protocols with widespread deployment—to uncover different protocol behaviors in regimes of intermittent connectivity. Chapter 4 designs a convergence layer to improve the utilization of link resources on the low-power radio technology IEEE 802.15.4 with the help of compression, fragmentation, and framing techniques. Chapter 5 approaches Quality of Service and fairness measures for NDN with a lightweight scheme to coordinate device resources. To cope with disruptive connectivity and producer mobility in wireless networks, Chapter 6 introduces an energy-friendly publish-subscribe extension for NDN.

Part II utilizes the experiences gained in Part I to construct a data-centric WoT that inherits information-centric principles. Chapter 8 validates the benefits of NDN in a firmware-rollout use case using a real testbed deployment with the objective to motivate interoperable integration levels of information-centric principles for the IP-based IoT. Chapter 9 compares the two security

[1] `https://gitlab.com/oscore/liboscore`

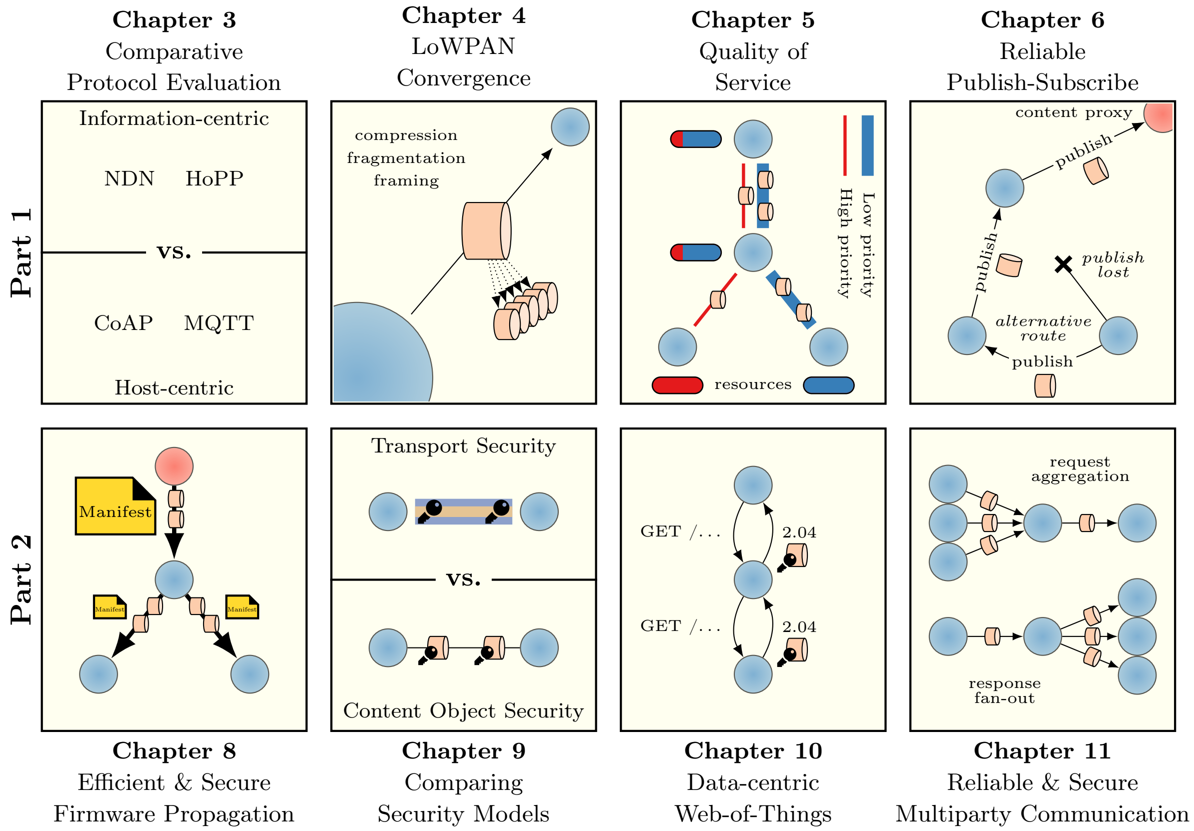


Figure 1.3: Overview of the parts and chapters included in this manuscript.

models (*i*) transport layer security, and (*ii*) content object security, and assesses their advantages and disadvantages for the IoT. Chapter 10 inspects the core protocol elements of CoAP to build a data-centric WoT, which operates exactly like NDN and adheres to its performance expectations. Since NDN inherently supports multi-source and multi-destination traffic flows, Chapter 11 provides extensions to acquire secured multiparty capabilities for the data-centric WoT.

Part I

Information-centric Networking for the Internet of Things

Chapter 2

Motivation and Problem Statement

2.1 Content Retrieval at the Internet Edge

The primary mission of many IoT deployments is to gather sensor data from numerous devices. For the most part, communication flows between nodes in an IoT edge network and a central cloud system, which processes the data to enable services for customers via the Internet. Sensor readings are usually small with a net size of a few bytes, and space-efficient binary encoding schemes, *e.g.,* CBOR [30], add additional structure to the data without an excessive overhead. While the actual packet has only a small impact on available link resources and many devices are regularly sleeping, the sheer number of individual sensor devices quickly multiplies the traffic volume, which degrades network performance in large deployment sites. In contrast to the small message retrievals of sensor data, IoT traffic can also consist of large packet transfers. Firmware updates cultivate a sustainable IoT, which adapts devices to evolving requirements, altering deployment aspects, and increasing security demands. The roll-out of firmware images happens less frequently than the retrieval of sensor data, but since image sizes are two to three orders of magnitude larger, their distribution occupies link and device resources for much longer periods. These peak loads can saturate large parts of a multi-hop LLN deployment.

The link-layer and upper layers employ actions to correct transmission failures, and in relaxed environments the likelihood is high that the link-layer successfully recovers a packet using only a single or very few retransmission attempts. In situations of increased network stress, however, all retransmissions on the link-layer are likely to fail, which triggers the much slower application-level retries. These retransmissions are performed end-to-end with host-centric transports, *i.e.,* they repeatedly traverse each node of a multi-hop path until a packet is successfully acknowledged by the origin. Especially in large-scale configurations with long path stretches, end-to-end traversals are challenged by error probabilities that accumulate destructively with each hop. At the same time, this recovery overhead introduces an increased utilization of the wireless medium, and hence impacts the cross traffic of neighboring nodes. This leads to a vicious cycle where data traffic fails due to recovery attempts, which then again raises further retransmission packets and more network stress.

A name-based, stateful forwarding and in-network caching as contributed by NDN [22] increase the robustness of content retrievals in regimes of low reliability by reducing the retransmission overhead. First, the hop-wise caching shortens the retransmission path, because content is potentially moved closer to a requesting node with each recovery attempt. Second, the stateful forwarding allows for aggregating requests and their retransmissions from multiple origins, which effectively reduces the number of actual packet (re-)transmissions. In Chapter 3, we inspect the feasibility of ICN protocols under harsh conditions as typical for the IoT in more detail, and experimentally derive strengths and weaknesses based on varying deployment scenarios.

2.2 ICN Convergence Layer for Low-Power Links

Popular IoT link-layer technologies are constrained in their performance to achieve a low energy consumption for IoT configurations. The IEEE 802.15.4. standard specifies a link throughput of 250 kbps or lower and an MTU of 127 bytes, of which roughly 80 bytes remain for the actual application data in realistic deployment scenarios with MAC layer security enabled. Established protocols on the Internet are not optimized for coding efficiency, but primarily evolved to be extensible, future-proof, and comprehensible by network architects and developers. They are designed for easy processing by general purpose Internet routers and hosts. This latter case requires header representations that are accessible without the need for extensive packing or compression operations. In contrast, link technologies in the IoT have scarce resources, and protocol adaptations are necessary to ensure correct protocol behaviors. Packets typically require several milliseconds for the actual transmission on these limited radios, while the intra-stack processing consumes time in the microseconds range [31]. This imbalance suggests that preliminary protocol transformations can even speed up packet deliveries, despite adding processing overhead.

The IP-world has created 6LoWPAN [3, 32] as a convergence layer that provides feasible frame encapsulation formats, packet header compression and link fragmentation for IPv6 packets in LLNs. The ICN world has not yet developed corresponding features for constrained environments, which results in suboptimal operations for protocol variants like CCNx and NDN due to various reasons: (*i*) These deployments utilize a very flexible protocol encoding using Time-Length-Value (TLV) header fields. This allows for the inclusion of variable-length name prefixes in packets, eases extensibility of the protocol features, and enables an unordered composition of header fields. However, the TLV usage in CCNx and NDN also results in a verbose header encoding, inflates packet sizes, and introduces redundancy, which is undesirable in constrained wireless regimes. Increased message sizes promote longer media utilization times, introduce higher latencies, and raise the likelihood of packet loss. Since MTUs are mainly low, size inflation reduces goodput rates and may even enforce packet fragmentation to deliver application payload. Reordering and restructuring the header with a stateless header compression may be highly advisable to achieve a more economical packet encoding. (*ii*) The name-based forwarding uses content names to match returning responses to open requests and to perform cache lookups on each forwarder. For this, names are included in both packet types. Human-readable names can grow quickly in length with refined naming schemes in realistic deployments. They may include device identifiers, key fingerprints, service classifiers, or follow other means of categorizing content. This redundant mirroring of names introduces a significant size overhead in returning responses and urges for alternative methods in regimes of limited resources. One point of reference for potential gains may be the stateful forwarding fabric of CCNx and NDN. In contrast to the end-to-end transport of the Internet, suitable data structures in ICN variants already maintain hop-wise state, and minor adaptations may easily allow for eliding redundant information within the scope of single request-response messages, or even across multiple distinct request round-trips.

In Chapter 4, we design ICN over Low-Power Wireless Personal Area Networks (ICNLoWPAN) [31, 33], a full-featured protocol convergence layer for an information-centric IoT, and provide an extensive evaluation to benchmark its impact on energy usage, packet size, message processing, and packet loss. It follows the design principles of 6LoWPAN, but further leverages the NDN potentials of stateful forwarding and employs highly efficient compression primitives. ICNLoWPAN further integrates into the existing 6LoWPAN dispatching framework, so it (*i*) reuses available features, such as the link fragmentation, and (*ii*) enables coexistence with 6LoWPAN networks.

2.3 Resource Coordination for the Information-centric IoT

Stateful forwarding and in-networking caching promise to increase reliability for data propagations in lossy regimes. General purpose deployments with fairly provisioned network resources can meet these additional state requirements, but the IoT edge struggles with intricate and resource-consuming solutions. Capacities in forwarding and caching are scarce on common IoT devices and can harm the efficient operation of native NDN deployments by quickly saturating available device resources. Unfavorable conditions may lead to the starvation of particular traffic flows. Considering the stateful nature of the forwarding plane, an uncontrolled, hop-wise state placement may even lead to full network disruptions for configurations where a path is gradually constructed towards the requested content object, but is never fully established due to saturated nodes en route. An unsuccessful construction of the request path is harmful, since state is occupied on each forwarder without a reasonable chance of prematurely consuming them by returning responses. Instead, these states may prevent other valid request paths from forming until they are garbage collected by relatively long request timeouts.

An overprovisioned network easily meets capacity requirements for expected traffic loads, but an abundant dimensioning of network resources is infeasible for the IoT. The coordination of available resources is desirable for controlling the degree of service quality in an insufficiently provisioned network. Quality of Service (QoS) in IP networks has been around for two decades, but so far has experienced remarkably little deployment. Its hesitant adoption is commonly understood to have two reasons: limited scalability (IntServ [34, 35]) and plain resource trading (DiffServ [36, 37])—both are often referred to as managed unfairness. While QoS in the IP world is mainly restricted to managing forwarding resources (link capacities and buffer spaces) [38], ICN variants offer additional resource dimensions such as in-network caches and forwarding states that can shape the network performance significantly.

A QoS approach like IntServ seems inappropriate for the constrained IoT case, since its signaling overhead scales with the number of distinct flows. Solutions based on differentiated services have a notably lower setup cost, but require global knowledge on traffic classes and their treatments. Nevertheless, the latter model seems more applicable to an information-centric IoT QoS scheme due to the lower protocol complexity and signaling overhead.

The first step to a successful coordination of resources is to identify packet flows and to map packets to particular traffic classes. The definition for information-centric traffic flows differs by design from the definition for host-centric traffic flows. While the prominent 5-tuple (IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, Protocol) distinctively identifies flows on the Internet, it is infeasible in the ICN world due to the missing concept of endpoint addresses. It stands to reason that name components in request and response messages can serve as flow classifiers. In contrast to differentiated service classes on the Internet, these variable-length components also allow for a hierarchically structured, unlimited expressiveness to enable numerous service classes with varying granularity. QoS schemes can leverage this to articulate flow treatments on top-level name components to affect a huge range of service classes, or on more specific name prefixes to define exceptional treatments.

In Chapter 5, we thoroughly explore the impact of a simple QoS management on NDN in the resource-constrained IoT [39]. NDN offers a range of resources that QoS schemes can leverage to enhance the performance for a subset of traffic flows. We follow the premise that network performance can be greatly improved by managing these resources, correlating them internally on a node, and also externally between nodes. The stateful packet processing of NDN allows for performing these external correlations without additional signaling overhead. Moreover, we accentuate the importance of fairness measures and integrate strategies to prevent network members from resource starvation. This ensures a continuous flow of prioritized traffic, while not sacrificing the performance of unprioritized traffic.

2.4 Publish-Subscribe for Mobile Wireless Networks

Publish-Subscribe has a big IoT use case and MQTT-SN is one popular publish-subscribe protocol designed for an efficient IoT communication. It employs a central message broker that routes publications to active subscriptions, and brings a loose coupling between sensor nodes and data consumers. The inherent support for point-to-multipoint messaging allows for a scalable communication, which liberates the constrained devices from managing group membership states. MQTT-SN fosters an asynchronous message processing, and eliminates the need for polling messages. Especially in event-driven systems, this reduces response times and delivery latencies. By controlling active topic subscriptions, IoT devices can further improve their sleep patterns, and collect publications on their terms from the central broker. In networks with many mobile nodes, such as hand-held devices and wearables, the loose coupling also reduces the complexity of managing mobility and routing states. Nevertheless, message transfers from publisher to the central broker, and from the broker to the subscriber still face the common issues of LLNs. Therefore, doubts arise whether an end-to-end publish-subscribe is the appropriate approach in disruption-prone environments of wirelessly connected things.

The content-specific interface to networking is considered a significant advantage of CCNx and NDN—consumer and producer applications can access content objects without any intermediary. Their stateful forwarding and in-network caching provide a hop-by-hop content retrieval, which also supports content consumer mobility: A consumer can simply repeat the request after it successfully moved to a new location. Producer mobility is a more delicate topic, though, since it requires global forwarding state updates. While NDN already decouples content provisioning from data producers in space, an additional decoupling in time and synchronization for handling disruptive content retrievals is desirable and attainable by a publish-subscribe layer.

Information-centric publish-subscribe networks have been proposed, and an early prominent candidate is PSIRP/PURSUIT [16]. Its central control paradigm, however, seems more suitable for LAN deployments that run, *e.g.,* a Software Defined Networking (SDN) architecture. Publish-subscribe schemes based on NDN, such as Content-based pub/sub [40] and COPSS [41], violate the loose coupling principle in their use of name-based routing or forwarding. Nichols [42] suggests broadcasting for pub-sub, which generally wastes energy and does not scale well with the number of network participants. Three challenges need addressing for creating a resilient content replication mechanism with low energy demands. (*i*) In the name-based forwarding of NDN, forwarding states grow with the number of published content and not with the number of network participants. This can lead to state explosion issues in large-scale IoT deployments with frequent content publications. (*ii*) Consumer and producer mobility as well as temporary network partitionings are prevalent and must be handled by any publish-subscribe solution. (*iii*) Constrained processing and memory resources necessitate a low implementation complexity, since all applications on a node—including the operating system and the network stack—compete for the limited resources and wasteful uses affect the device operability.

Facing the current state of the art, an exploration of the problem space for an IoT-based, information-centric publish-subscribe networking with a particular focus on mobile and intermittently connected sensors and actuators seems necessary. In Chapter 6, we introduce HoP-and-Pull (HoPP) [43] to take up this challenge and to seek for a solution that qualifies for real-world deployments with resource-constrained characteristics. HoPP makes the common assumption that nodes form a stub network and connect to the outside by one or several gateways; by exploiting the lean routing protocol PANINI [44], HoPP can build on prefix-specific default routes instead of broadcasting, and thus requires only minimal forwarding states independently of the number of nodes in the network. For the interior routing, nodes arrange according to one or many subnetwork prefixes (*e.g.,* `/lighting`). Distinguished nodes serve as Content Proxies (CPs), which are typically more stable and more powerful gateways or other infrastructural entities. These Content Proxies take the role of data caches and persistent access points. They will be reachable throughout the network by default routes, unless temporary partitioning occurs. A CP can serve multiple local prefixes, but a local prefix may also belong to many CPs. The latter scenario will lead to replicated caching with higher and faster data availability.

Chapter 3

Potentials of ICN for the Internet of Things

Abstract

Common use cases in the Industrial Internet of Things (IIoT) deploy massive amounts of sensors and actuators that communicate with each other or to a remote cloud. While they form too large and too volatile networks to run on ultra-reliable, time-synchronized low-latency channels, participants still require reliability and latency guaranties. We elaborate this for safety-critical use cases. This chapter focuses on the effects of networking protocols for industrial communication services. It analyzes and compares the traditional Message Queuing Telemetry Transport for Sensor Networks (MQTT-SN) with the Constrained Application Protocol (CoAP) as a current IETF recommendation, and also with emerging Information-centric Networking (ICN) approaches, which are ready for deployment. Our findings indicate a rather diverse picture with a large dependence on deployment: Publish-subscribe protocols are more versatile, whereas ICN protocols are more robust in multi-hop environments. MQTT-SN competitively claims resources on congested links, while CoAP politely coexists on the price of its performance.

3.1 Industrial IoT Use Cases

The Industrial Internet of Things (IIoT) revolutionizes how processes in industrial environments are controlled. It makes use of local data aggregation, processing on the edge, and cloud computing to refine and optimize process controls. Here, infrastructure such as sensors and actuators are interconnected. Applications range from aggregating locally stored log data (reporting) to sensing and raising alarms if thresholds are undercut or exceeded (monitoring) and even intervening processes with regulating actuators (controlling) [45]. The aggregation of log data has to be reliable and secure. Reliable in a sense that all existing data needs to be transferred to a designated device, secure within respect to the integrity and authenticity of the data. While the sensing and propagation of non-critical data in-network has modest timing requirements, critical data such as alarms and control commands need to be forwarded and disseminated to relevant parties with low latency.

Among others, use cases include monitoring and reporting of environmental and vital data

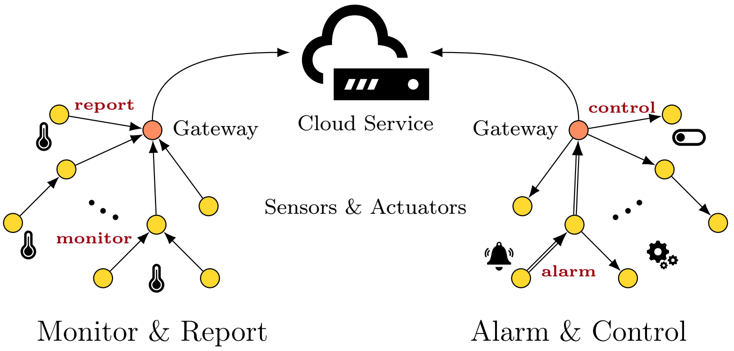


Figure 3.1: Communication flows in IIoT environments.

of workers in harsh environments (*e.g.,* early responders), process regulation in manufacturing industries (*e.g.,* chemicals, gas, oil, minerals), and even factory automation using with control of robots (*e.g.,* assembly lines) [46]. An overview of the communication flows as typical for industrial environments is visualized in Figure 3.1.

Safety-critical Environments. Safety-critical environments clearly benefit from the Industry 4.0 paradigm, *i.e.,* networking the control components, because here advanced communication interfaces do not only improve manufacturing processes but may also help to save lives. Concrete examples are industrial processing plants or refineries. Typically, a mix of personal mobile and fixed gas detectors are used to sense the environment for possible leaks of hazardous or combustible gases. Often multiple teams of workers perform maintenance tasks in designated areas, in which every worker is equipped with a gas detector. In addition, fixed gas detectors are deployed on critical infrastructure, which support the maintenance tasks of workers. Detection of a dangerous level of gas switches the gas detectors to alarm mode. In a networked scenario, each detector sends a message to a centralized safety monitoring application that runs either locally or in the (edge) cloud. Based on new alarm information, the safety manager decides whether to preemptively evacuate close-by areas.

Industrial Control Systems. Control systems are widely deployed in industrial process automation, where they continuously monitor flows, and in factory automation, where they mainly deal with discrete on/off signals of machines like robots. Continuous monitoring periodically transmits process values and directly adjusts control of actuators such as valves or pumps in a closed loop. In contrast, discrete control signals are event driven (*e.g.,* generated from a relay after a robot action) and require individual reactions, which are not stabilized by corrective periodic updates. Control events may be critical and consequently more sensitive to signal delays or losses.

Deployment of sensors and actuators in industrial production environments is likewise harsh. Plants often undergo unpredictable variations in the environment (*e.g.,* temperature, humidity, vibrations), in the radio regime (*e.g.,* cross traffic, reflections from moving metal objects, steam emittance from machines), and energy-wise. Many field devices operate on batteries and may need to survive periods from days to months between recharging and general overhauls.

Requirements. From the networking perspective, the following requirements have to be satisfied to fully support these industrial safety use cases. The most important requirement is latency for alarm messages from the detector to the safety application and for evacuation requests in the reverse direction. Being able to react quickly to safety relevant incidents is crucial to contain and resolve dangerous conditions. The ANSI/ISA-100.11a-2011 standard [47] defines latency requirements for three traffic categories in industrial process automation applications:

1. *Safety* traffic indicates emergency and requires a maximum of 10 ms delay in a deterministic fashion.

2. *Control* traffic is often but not always critical and depends on its application context, latencies between 10 and 100 ms are sufficient.

3. *Monitoring* traffic is used for maintenance and should deliver messages within 100 ms on average.

Additionally, lost messages may lead to undetected alarms in the safety monitoring software and, hence, a *high reliability* is crucial.

When not in alarm mode, detectors log their sensor readings on the device and send their status once per minute. This frequency increases when a detector changes to alarm state, since regulations stipulate that the sensor readings need to be logged at least once per second. Those logs are required for any investigation following up the particular gas alarm incident. Thus, it is desirable to *send data at very low frequency* to the centralized safety application.

From an operational perspective, the network architecture should allow for the deployment of a flexible ecosystem, which enables private as well as open networks.

Challenges. Meeting these requirements is challenging in harsh industrial environments, where time-slotting traffic schedules are difficult to deploy. Workers are constantly moving, and path-loss and shadowing effects appear due to the massive amounts of steel used in processing plants. In addition, there may be uncoordinated side channel traffic initiated by co-located systems from different manufacturers, which is particularly harmful for synchronized communication channels as defined in IEEE 802.15.4e (TSCH) [2] deployments [48, 49]. In case of larger incidents, in which several hundred or thousand detectors send alarm notifications, coping with network traffic is even more challenging. And finally, some industrial areas are so remote that network coverage provided by technologies such as cellular is very poor or non-existing.

On the upside, monitoring the complete gas detector status typically fits in less than one kilobyte of data. Thus, the required available data rate is very low.

Potentials of 5G. A key building block for a successful IIoT is 5G [50]. Massive machine type communication (mMTC) provides a narrowband Internet access for sensing, actuating, and monitoring devices. The ultra-reliable low latency communication (URLLC) in 5G will provide sub-millisecond latency communication, which is essential for dedicated devices in process control. Additionally, allowing industrial customers to operate their private 5G-based networks provides the chance to close coverage gaps in remote areas. These private networks can then be interconnected with a mobile carrier's network. Finally, 5G opens the scene for a data-centric network core, which may help to increase reliability in constrained and lossy environments.

Having a promising network access architecture such as 5G in place still requires efficient protocols on top. The current IIoT ecosystem proposes several competing solutions. These protocols require careful evaluation with respect to resource allocation, convergence problems, and coexistence scenarios, in particular in the context of a safety-critical Industry 4.0.

3.2 Networking Protocols for Industry 4.0

Domain-specific protocols in the IIoT include Zigbee, ISA100.11a, and WirelessHART [51, 46], all of which specify a full protocol stack which can be configured to application requirements. This is done by a centralized instance, usually called a network manager. The network- and transport layers deal with IP connectivity on a backbone router whereas routing between constrained devices is implemented in a proprietary fashion directly on top of the MAC layer.

Standard IoT networking protocols to handle massive volumes of heterogeneous data flows are CoAP and MQTT on the application layer in the current Internet, and information-centric (or data-centric) networking for the next generation IIoT. The latter provides higher layer services known from the application layer, such as naming and caching, directly on top of the data link layer. In this section, we briefly give technical background to common link technologies in the IIoT, and provide a qualitative comparison of the core protocols CoAP, MQTT and ICN.

3.2.1 Common Link Layers for the IIoT

Industrial protocols to handle data flows of sensors and actuators heavily rely on the MAC at its link layer, which we briefly discuss here. The popular 802.15.4 family is a characteristic example of lossy local area wireless transmission at minimal energy. We base our experimental work on 802.15.4 with non-slotted media access to provide robust transmissions and neutral performance impact, for the absence of time schedules.

IEEE 802.15.4-based technologies. Many short range wireless solutions in the IIoT are built on IEEE 802.15.4, which specifies low-power and low-rate physical layers and media access control. Prominent examples are Zigbee, ISA100.11a, ore WirelessHART. The PHY in most deployments operates on the 2.4 GHz band and applies a simple O-QPSK (Quadrature Phase Shift Keying) modulation. Symbols are spread in the code domain to operate on a direct-sequence spread spectrum (DSSS). This increases resistance against narrow-band interference.

We distinguish two classes of media access with this technology: (*i*) time-slotted and (*ii*) non-slotted multiple access. The former reduces energy consumption, though, its performance is heavily affected by the scheduling logic upfront. Furthermore, network synchronization is susceptible to interference. In contrast, non-slotted access omits scheduling and exploits carrier sensing to avoid collisions.

Wireless media is susceptible to eavesdropping, and security between neighbor nodes is provided by the 802.15.4 MAC. Hence, higher layer security is still required to achieve security on data domain. 802.15.4 specifies eight levels of protection which reflect increasing security strengths to achieve data privacy, integrity, and authenticity. Data encryption and message integrity codes utilize AES with 128 Bit keys, though, provisioning of keys between peers needs to be handled by the upper layer, or manually during deployment. In addition, access control lists exclude frames that are received from untrusted nodes and hence, could be malicious. It is noteworthy, though, that bare 802.15.4 is still vulnerable to a number of attacks [52, 53].

All three standards mentioned above utilize the time-slotted channel hopping mode of the IEEE 802.15.4e specification to guarantee link resources. This type of time- and frequency multiplex requires coordination among nodes to synchronize to a schedule, and to grant resource access. Hence, it adds signaling overhead, especially for sporadic and asynchronous data. The slot mode, however, enables device sleep cycles to save energy. The IETF adopted 6TiSCH [54, 55] as an open standard solution that bases on the above mentioned protocols and enables IPv6 connectivity on constrained nodes themselves. Due to central coordination and susceptibility to side-channel interference [48, 56], however, TiSCH-type link layers do not meet the use cases of uncoordinated deployment in harsh industrial environments. We therefore base our experimental work on the contention-based and grant-free CSMA/CA mode of IEEE802.15.4 and concentrate on the performance impacts of the higher layers.

Novel, non-orthogonal technologies. Orthogonal access schemes like 802.15.4 as presented above, are key to current wireless systems, however, the orthogonality criterion limits the number of users. Consequently, mMTC platforms advance in modulation and multiplexing by introducing non-orthogonal schemes to the space, time, frequency, or code domain [57, 58]. This allows for resource overloading to extend the number of simultaneous users but also increases receiver complexity. Sparse code multiple access (SCMA) is a core technique in 5G systems which operates in the code domain to enable overloading. SCMA maps data-streams to non-orthogonal code streams. Codewords of multiple SCMA-layers are combined and transmitted over OFDMA (orthogonal frequency-division multiple access), a multi-carrier technique with time slotted access. Space division is achieved by traditional cell clustering and advanced with antenna beamforming to reduce cell overlap, and thus, to increase resource re-utilization. Hence, 5G extends media access in four dimensions: code, frequency, time, and space.

Table 3.1: Comparison of CoAP, MQTT, and ICN protocols. CoAP and MQTT support reliability only in confirmable mode (c) and QoS levels 1 and 2 (Q1, Q2).

	Current IoT Protocols					**ICN Protocols**	
	CoAP			MQTT	MQTT-SN	NDN	HoPP
	PUT	GET	Observe				
Transport	UDP	UDP	UDP	TCP	UDP	n/a	n/a
Pub/Sub	✘	✘	✓	✓	✓	✘	✓
Push	✓	✘	✓	✓	✓	✘	✘
Pull	✘	✓	✘	✘	✘	✓	✓
Flow Control	✘	✘	✘	✓	✘	✓	✓
Reliability	(c)	(c)	✘	(Q1, Q2)	(Q1, Q2)	✓	✓
Security Mechanism	transport / content	transport / content	transport / content	transport	transport	content	content
End-to-end Protection	(✓)	(✓)	(✓)	✘	✘	✓	✓

3.2.2 Common Application Layer Protocols for the IIoT

The IETF solution, CoAP. The Constrained Application Protocol (CoAP) [4] aims for replacing HTTP to enable M2M communication between constrained nodes. In contrast to HTTP, CoAP is able to run on top of UDP and introduces a lean transactional messaging layer to compensate for the connectionless transport. CoAP provides a more compact header structure than HTTP. It currently supports three communication primitives: (*i*) pull, (*ii*) push, and (*iii*) observe. Pull implements the common request response communication pattern. However, as IoT scenarios also include the proactive communication of unscheduled state changes, CoAP was extended to support pushing new events to its peers. Still, this does not allow for publish-subscribe scenarios when producer and consumer are decoupled in time and data is not yet available at the request. The support for delayed data delivery in publish-subscribe was specified in CoAP observe [59]. Here, clients can signal interest in observing data, which implies that a CoAP server delivers data as soon as available and maintains state until clients explicitly unsubscribe. The default approach to reinforce communication channels in CoAP deployments is to use (datagram) transport layer security (D)TLS [60, 25]. OSCORE [26] is a recent addendum to the CoAP specification and allows for securing content objects on the application layer, in addition to any transport protection.

CoAP is the IETF standard for implementing data transfer on the application layer in the future Industrial Internet of Things. Currently, several implementations exist, as well as early adoption in a few selected products and deployments.

The well-deployed solution, MQTT. The Message Queue Telemetry Transport (MQTT) [5] was designed as a publish-subscribe messaging protocol between clients and brokers. Clients can publish content, subscribe to content, or both. Servers (commonly called *broker*) distribute messages between publishing and subscribing clients. It is worth noting that the protocol is symmetric: Clients as well as brokers can be sender and receiver when MQTT delivers application messages.

MQTT is considered a lightweight protocol for two reasons. First, it provides a lean header structure, which reduces packet parsing and makes it suitable for constrained devices with low energy resources. Second, it is easy to implement. In its simplest form, MQTT offloads reliability support completely onto TCP.

To provide flexible Quality of Service on top of the underlying transport, MQTT defines three QoS levels. *QoS 0* implements unacknowledged data transfer. An MQTT receiver gets a message at most once, depending on the capabilities of the underlying network, as there is no retransmission at the application layer. *QoS 1* guarantees that a message is delivered at least once. Based on timeouts, an MQTT sender will retransmit application messages when an acknowledgment is missing. *QoS 2* ensures that a message is received exactly once, to avoid packet loss or processing of duplicates at the MQTT receiver side. This requires a two-step acknowledgment process and more state at both sides.

To adapt MQTT to constrained networks which are based on low data rates and very small packet lengths such as in 802.15.4, MQTT-SN [6] is specified. Header complexity is reduced by replacing topic strings with topic IDs, to identify content. In contrast to MQTT, MQTT-SN is able to run on top of UDP. It still supports all QoS levels but does not inherit any reliability property from the transport layer.

MQTT provides optional header fields during the establishment of connections to authenticate with a broker, but most other responsibilities, such as encrypting and authenticating published data, are relayed to the application. The transport is commonly protected using transport layer security (TLS) for the TCP-based MQTT, and the datagram variant DTLS for MQTT-SN. The specification provides implementation notes and guidance for a secured deployment in the protocol specification [5, Section 5].

3.2.3 Upcoming Data-centric Networking Layers

Information-centric networking (ICN) implements the vision of a native data-centric Internet. The most active approach is named-data networking (NDN). The core **NDN** protocol [22] combines name-based routing with stateful forwarding to deploy a request response scheme on the network layer. Any consumer can request named data using so-called Interest messages, which are forwarded towards publishers. Data is subsequently delivered along a trail of reverse path forwarding states, starting either from the original publisher or the first in-network cache that can provide the requested data. As an important feature, data will only be delivered to those who requested the data. This means that data must be (individually) named at the Interest request and that yet unavailable data requires repeating Interests until the application receives the data. Due to the comprehensive use of on-path caches and the stateful forwarding fabric, the concept of endpoints is of negligible importance for NDN deployments. Thus, these regimes allow for an orthogonal approach of delivering autonomously verifiable content objects independently of location and communication endpoints.

Several publish-subscribe extensions have been proposed for NDN [41, 61, 62] to provide further decoupling of consumers and data sources. **HoP and Pull (HoPP)** [62] is a lightweight variant we previously developed to provide a publish-subscribe system for constrained IIoT deployments based on ICN/NDN principles. A constrained publisher announces a name towards a content proxy to trigger content requests and to replicate the data towards a content proxy (or broker). Forwarding nodes on the path between publisher and content proxy hop-wise request content for this name by using common Interest and data messages. A content subscriber in HoPP behaves almost like any content requester in NDN and issues a regular Interest request towards the content proxy. However, in contrast to NDN (*i*) a subscriber cannot extract content names from its forwarding information base (FIB), since FIBs only contain default routes [44], but uses application-specific topic tables instead; (*ii*) it does not expect an immediate reply, but issues Interests with extended lifetimes. HoPP enables rapid communication of unscheduled data events. It operates at a similar timescale as push protocols without actually pushing data.

3.2.4 Qualitative Comparison of Protocols

Key properties of the three protocol families CoAP, MQTT, and NDN and its variants are compared in Table 3.1. Specialized properties of the different approaches become apparent: Every protocol variant features distinct capabilities. Notably in the IoT, where TCP (aka generic MQTT) is unavailable, the pull-based NDN and NDN-HoPP are the only protocols admitting flow control and reliability as a generic service. Further, low-power deployments show a growing demand for application gateways to perform protocol conversions and changing the transport, *e.g.,* from UDP to TCP. These operations naturally break the end-to-end principle [63] at gateways, terminate any transport security, and therefore render the communication between constrained IoT devices and cloud services vulnerable to interception attacks [64, 65]. The NDN family of protocols and CoAP with OSCORE protection can guarantee security properties to remain intact beyond protocol conversions [66]. In addition, content object security enables multicast and multi-homing capabilities for the IoT, while these are hardly feasible to deploy with transport protection due to the tight endpoint binding. This especially affects setups that experience device mobility and frequent network disruptions.

To give a first estimate of the different protocol complexities, we compare the sizes of the message types for each protocol in Figure 3.2. Most of the protocols need nearly the same amount of data. CoAP observe (CoAP OBS) exhibits the lowest complexity but does not acknowledge. A single registration is sufficient to receive subsequent data published under the same name. HoPP, on the other hand, introduces overall the largest packet size as it introduces

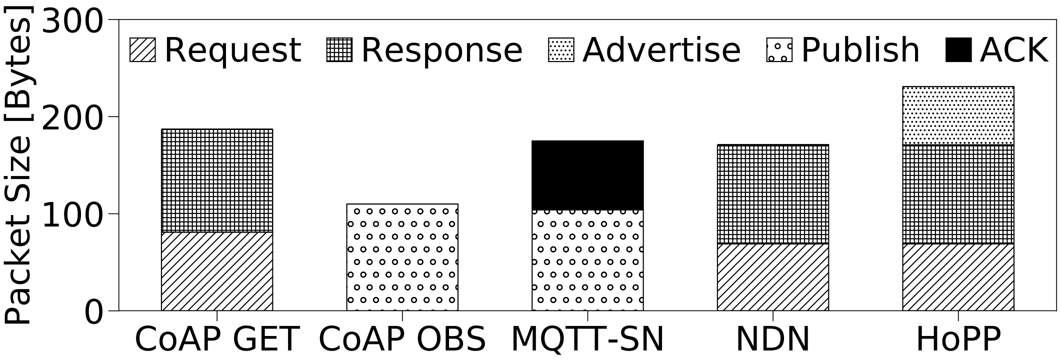


Figure 3.2: Packet sizes in bytes for each protocol.

name advertising on the data plane. We elaborate the scenario and give a comprehensive performance evaluation in the next sections.

3.3 An Environment for Assessing Industrial M2M Networks

Common deployments in the IIoT consist of stub networks that are single-hop in areas of dense infrastructure, but may also be multi-hop in widespread facilities such as oil refineries or platforms. Traffic flows from or to the edge nodes in three patterns: (*i*) scheduled periodic sensor readings, (*ii*) unscheduled and uncoordinated data updates, or (*iii*) on demand notifications or alerting. It is worth noting that the different protocol properties (*e.g.,* pub-sub versus request-response) meet these alternating demands with varying success. In the following, we present a testing environment consisting of software, a real-world testbed, and relevant scenarios that approximates the characteristics of massive M2M networks for embedded devices.

3.3.1 Software Platforms

On the constrained nodes, all of our experiments are based on the RIOT operating system [28] version `2018.01`. To analyze CoAP, MQTT-SN, and NDN we use `gCoAP`, `Asymcute`, and `CCN-lite` respectively. All three protocol implementations are part of the common RIOT release and thus reflect typical software components used in low-end IoT scenarios.

Brokers or gateways are deployed on Linux systems within the testbed infrastructure. To support an MQTT-SN broker and a CoAP observe client, we used `aiocoap` version 0.3 and `mosquitto.rsmb` version 1.3.0.2. Both are popular open source implementations in this context.

3.3.2 Testbed

We conduct our experiments in the FIT IoT-LAB[1] testbed. The hardware platform consists of typical class 2 devices [1] and features an ARM Cortex-M3 MCU with 64 kB of RAM and

[1] `http://www.iot-lab.info/`

512 kB of ROM. Each device is equipped with an Atmel AT86RF231 transceiver to operate an IEEE 802.15.4 radio. The gateway runs on a Cortex-A8 node, which is more powerful than the M3 edge nodes. Every node in the testbed is monitored by a control node which allows for parallel radio sniffing without misusing transceivers of M3 devices.

The testbed provides access to several sites with varying properties. We perform our experiments on different sites, to analyze single-hop as well as multi-hop scenarios.

Single-hop topology: The *Paris* site consists of approximately 70 nodes, which are within the same radio range. We choose two arbitrary nodes and run all single-hop experiments on them. One node is a content producer, the other node acts as consumer (gateway/broker).

Multi-hop topology: The *Grenoble* site consists of approximately 350 nodes spread evenly in the Inria Grenoble building. We choose 50 M3 nodes (low-end device) and one A8 node (gateway/broker) arbitrarily and run all multi-hop experiments on them. All low-end devices operate as content producers. In our CoAP and MQTT experiments, we use RPL to build and maintain the routing topology across all nodes. For NDN-based experiments we build analogous tree topologies. Typical path lengths are four to six hops.

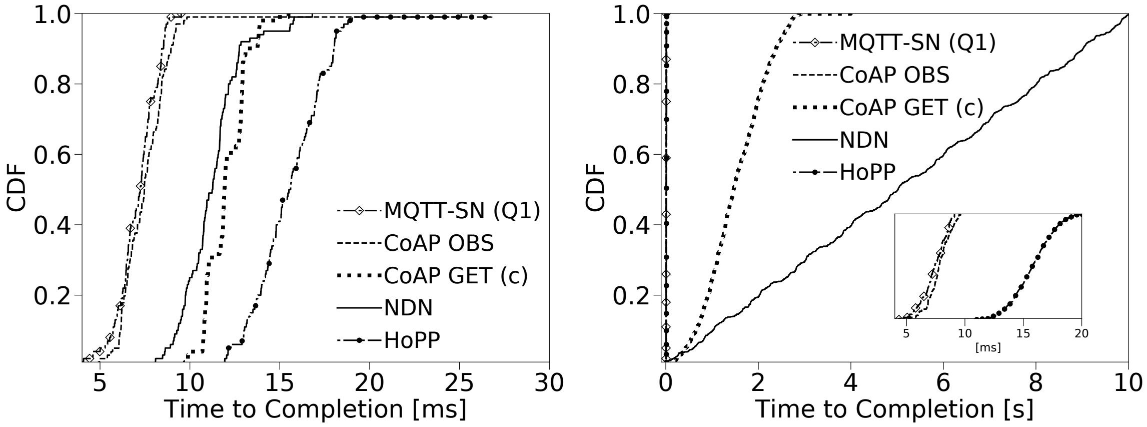
Two-hop topology with cross-traffic: We choose three M3 nodes that are arranged in a line topology within the *Grenoble* site. One node acts as a consumer, another node serves as a producer and the last node is a forwarder in between. Additionally, we deploy a fourth node acting as a cross-traffic generator in the vicinity of our forwarder.

3.3.3 Scenarios and Parameters

We align all experiments with respect to the configurations of retransmissions and timeouts to ensure comparability among protocols. All protocols employ the same retransmission strategy: In case of failures, each node waits 2 seconds before retransmitting the original application or control data. For NDN and HoPP, retransmissions are performed hop-by-hop, while CoAP and MQTT retransmit from end to end. At most 4 retransmissions will occur for each data item. Interest lifetimes are configured to 10 seconds for NDN based protocols to limit PIT memory consumption. We repeat each experiment 1,000 times.

To accommodate all 50 nodes in the routing topology, the FIB sizes have been adjusted accordingly on each constrained node. For CoAP and MQTT, this translates in our IPv6 scenario to a FIB size of 50 entries with roughly 32 bytes each. In our NDN scenarios, each node owns a unique prefix of the form $/\rho_i$ with a length of 24 bytes. The next-hop face of each FIB entry points to the 8-byte IEEE 802.15.4 link-layer address. In total, this setup yields comparable size requirements for all scenarios.

In the NDN scenarios, we use unique content names prefixed by $/\rho_i$ with incremental local packet counters. CoAP works without unique names but uses common URIs. The MQTT-SN protocols register a common topic name, similar to CoAP, and publish under a unique topic ID



(a) Scheduled publishing at 1 s interval (b) Unscheduled random publishing in 1s ... 3s

Figure 3.3: Time to content arrival in a single-hop topology.

thereafter. In all scenarios, the data is of the same JSON format consisting of a unique identifier and a sensor value attribute. These short messages can be accommodated by the link layer and do not require fragmentation. It is noteworthy that we neither apply header compression in the IP [67] nor in the NDN world [68].

3.4 The Impact of Topology in Massive Deployment

The objective of this work is to quantify the impact of network protocols on IIoT communication systems. With this goal in mind, we deploy the different publish-subscribe and request-response protocols in the same physical environment and compare their operational properties as well as their performance results. Evaluation metrics focus on reliability and timeliness of the data delivery, which are critical in the low power lossy environment of these systems. Additionally, we study link stress and resource efficiency of the constrained data flows. We start our analysis by comparing single- versus multi-hop topologies.

3.4.1 Single-Hop Topology

Protocol performances are first evaluated in a single-hop topology at the Paris testbed of IoT-LAB. In agreement with the requirements of our industrial use case, we perform a periodically scheduled publishing at every second, and a randomized, unscheduled publishing. We measure the time until content arrives at the consumer. The results are summarized in CDFs as functions of packet transfer time, see Figure 3.3.

In the case of scheduled traffic, all protocols successfully deliver data packets within short, similar times as shown in Figure 3.3a. Lightly visible steps in the CDFs indicate retransmissions on layer 2, which occur on the same timescale of milliseconds. Naturally, the protocols that push data (MQTT, CoAP OBS) react quicker than request-response schemes. As a pull-based publish-subscribe scheme, HoPP performs slowest, as it initiates hop-wise data transfers on request.

Our second evaluation addresses the common IoT use case of publishing data at irregular intervals. This is the typical pattern for observing third party actions (*e.g.,* alarms), or largely uncoordinated sensing environments. The publish-subscribe protocols naturally serve these application needs. We quantify the behavior of the request-based protocols in practice and chose the moderate setting of publishing content every two seconds on average. Publishing is uniformly distributed in the interval of $[1\,s, \ldots 3\,s]$. The protocols CoAP and NDN request the content periodically every second so that updates are not lost.

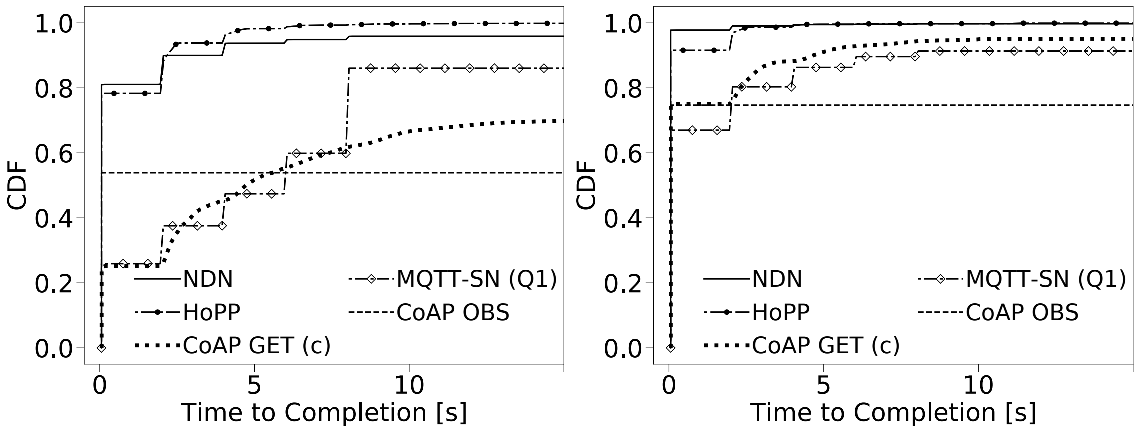
Figure 3.3b visualizes content delivery times for unscheduled publishing and reveal a diverse picture. CoAP GET and NDN now operate on a timescale of seconds, while the publish-subscribe protocols continues to complete in the unaltered range of $10\,ms$ without additional protocol operations – the unsurprising outcome of content triggers. CoAP requests content using a common name with the result of likely duplicate content transmissions. On average, CoAP needs two requests to retrieve fresh content with the expected average delay of $\approx 2\,s$ and a corresponding polling overhead of 200 %, see Figure 3.3b. In contrast, NDN exhibits lower overhead, as Interests are locally managed at the PIT and only retransmitted after state timeout. Issuing Interests at a higher rate than content arrival, however, leads to an accumulation of open states in the PIT. As resources on the constrained nodes are tightly bound, the PIT limits are quickly reached and can be only met by either *discarding* newly arriving Interests, or by *overwriting pending Interest state.* Both countermeasures delay content delivery, as can be seen in Figure 3.3b.

3.4.2 Multi-Hop Topology

We now consider the more challenging use case of mixed communication in multi-hop topologies: 50 nodes exchange content that is published every 5 or 30 seconds in an uncoordinated manner. Repeated experiments were performed on the Grenoble testbed with tree topologies of routing depths varying from four to six hops.

First, we examine the temporal distributions from content publishing to arrival in analogy to the single-hop cases. Figure 3.4 combines the results for all protocols, as well as both publishing rates. The overall results reveal a much slower and less reliable protocol behavior than could be expected from the single-hop values in Figure 3.3. Graphs reflect the common experience in low power multi-hop environments that interferences and individual error probabilities accumulate in a destructive manner.

The IP-based protocols, which operate in an end-to-end paradigm, now all fail in delivering data, the publish-subscribe protocols CoAP OBS and MQTT-SN representing the lower end. Widespread temporal distributions indicate repeated retransmissions on the network layer that operate on the scale of many seconds and still cannot compensate losses. In contrast, the hop-



(a) 5 s publishing interval

(b) 30 s publishing interval

Figure 3.4: Time to content arrival in multi-hop topologies of 50 nodes for publish-subscribe and request-response protocols at different publishing intervals.

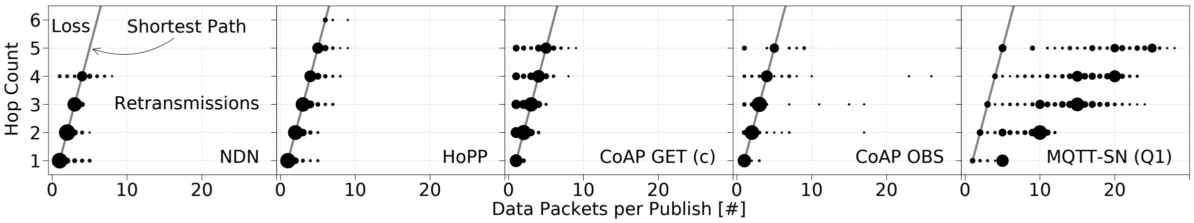


Figure 3.5: Link traversal vs. shortest path for a 30 s publishing interval. The scatterplots reveal the link stress with dot sizes proportional to event multiplicity.

by-hop nature of the ICN protocols demonstrates its robustness in these harsh environments. The publish-subscribe protocol HoPP quickly reaches 100 % success in data transfer – 80 % (Figure 3.4a) resp. 95 % (Figure 3.4b) of data units arrive within milliseconds and without any network layer retransmission. The performance of the plain NDN also shows decent results both in promptness and reliability, even though 5 % of data chunks remain lost in the fast publishing scenario of 5 s.

Second, we focus on the link utilization. We measure all individual paths that each unique data packet traveled on its destination from source to sink, and contrast the results with the corresponding shortest possible path. Results are visualized as scatterplots in Figure 3.5. Each dot represents one or several events, the dot size is drawn proportionally to event multiplicities. Solid lines indicate the shortest paths, while events left of the line represent failures (traversal shorter than the shortest path). Right of the solid diagonal retransmissions are counted.

The ideal protocol performance is situated on the diagonal line with all data traversing each link only once on the shortest path. This ideal behavior is most closely approximated by the NDN core and the NDN-HoPP protocols. A largely contrasting performance can be seen from

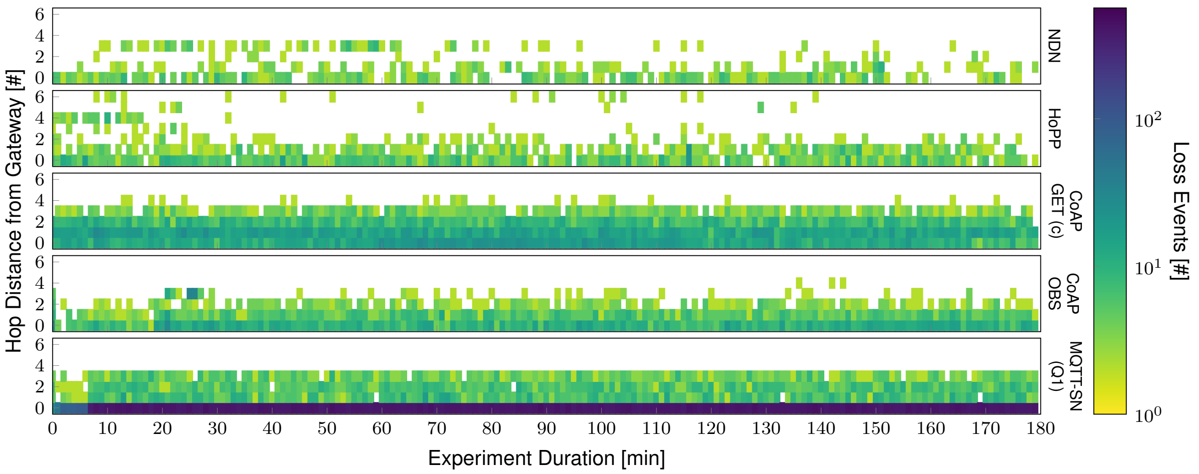


Figure 3.6: Loss count at links as a function of experiment time and hop distance. Cells show the loss intensity per minute for a 30 s publishing interval.

the reliable IP protocols MQTT-SN (Q1), which exhibits huge numbers of retransmissions. These retransmissions stress an exhausted link even further and stimulate cascading failures. The CoAP protocol variants behave more network friendly, thereby accumulating loss in a polite fashion.

We further question the details of packet loss and count the transmission failures on each link during the experiment. Figure 3.6 displays the number of packets lost in one minute as a function of time and hop distance from the gateway. Note that in this analysis every packet lost on some link is counted, no matter whether the retransmission mechanisms on the different layers can compensate this loss. An overall successful packet transfer in this analysis can thus account for many loss events on intermediate links. Frequent losses indicate a less effective link utilization by the network protocol.

It is common for this convergecast scenario that loss intensity increases toward the gateway, which serves as the root of the routing tree. Here packets accumulate on the last hops, why link exhaustion, collisions, and buffer drops increase. The effective success rate of packet traversal is largely influenced by the flow properties (*i.e.,* bursts versus balanced flows) as shaped by the networking protocol. In this, the protocol behaviors largely differ and lead to diverging results. The ICN protocols NDN and HoPP in Figure 3.6 show a more random distribution of small losses, which is typical for wireless interference and can be compensated by local retransmissions. In contrast, the IP-based protocols all suffer from more intense losses close to the gateway—loss of IP packets exceeds ICN loss by factors between 10 and 100. Only CoAP OBS looses moderately, because CoAP retransmissions are not active in this protocol variant and the total number of packets remains lower.

Compared to the confirmable CoAP GET configuration, MQTT-SN exhibits less loss events on the links farther away from the source, because of its more compact packet encoding. Extreme

Protocol	μ [mJ]	σ [mJ]	min [mJ]	Q1 [mJ]	median [mJ]	Q3 [mJ]	max [mJ]	sum [mJ]
NDN	98.99	213.96	23.66	23.66	23.88	70.98	1,243.54	4,949.50
HoPP	167.33	271.50	34.69	37.55	44.87	158.37	1,494.29	8,534.27
CoAP GET (c)	151.61	293.72	25.62	27.94	29.54	82.26	1,411.53	7,732.13
CoAP OBS	55.78	89.66	10.59	12.88	20.17	42.92	371.84	2,844.80
MQTT-SN (Q1)	245.66	394.63	65.61	68.66	74.91	183.10	1,915.61	12,529.12

Table 3.2: Statistical key properties of nodal energy expenditiures w.r.t. radio transceiver operations, *i.e.,* actively sending and receiving. Values calculate over the experiment duration for our protocol selection configured with a 30 s publishing interval. Q1 and Q3 represent the first (25%) and third (75%) quartile, respectively.

loss values show up at the source for MQTT-SN, however, due to its uncoordinated, bursty retransmissions. These effects are amplified in the multi-hop tree topology as the total network traffic accumulates towards the few links that directly connect to the gateway node. This explains the details behind the large transmission numbers seen in Figure 3.5.

Next, we comparatively examine the nodal energy consumption as a function of time throughout an experiment duration of ≈60 minutes for each deployment in our protocol selection. In the typical IoT scenario of acquiring and distributing sensor readings, energy expenditures due to computational efforts usually remain within tolerable limits. Radio activities, on the other hand, dominantly drive energy demands when receiving and transmitting data over the air. To concentrate on power expenses based on protocol characteristics, such as packet sizes and the quality of corrective actions, we only measure energy levels for actual radio operations. Consequently, we disregard expenses due to actively listening on the radio in our calculations, since this part of the equation decreases substantially in proper deployments with correct duty cycling and utilizing low-power modes. We obtain the power consumption levels for transmitting and receiving from the Atmel AT86RF231 [69] data sheet and convert nodal packet statistics into appropriate energy expenditures.

Table 3.2 compiles the statistical key properties for the nodal energy expenses of the 50 nodes in our multi-hop setup with a publishing interval of 30 s. Principally, maximum values represent energy levels of gateway nodes, since packets naturally accumulate there due to the convergecast setup. The 25% (Q1) and 75% (Q3) quartiles roughly illustrate the energy distributions. We note that nodes closer to the gateway are much more engaged with forwarding duties and experience additional radio activities when compared to leaf nodes. Thus, these nodes generally position towards the higher end of the distribution.

The average consumption for a single node greatly varies between the selected configurations, but agrees with our previous conclusions in Figure 3.5 and Figure 3.6. CoAP OBS displays the lowest average expense with ≈55 mJ per node, which is expected due its push-based nature

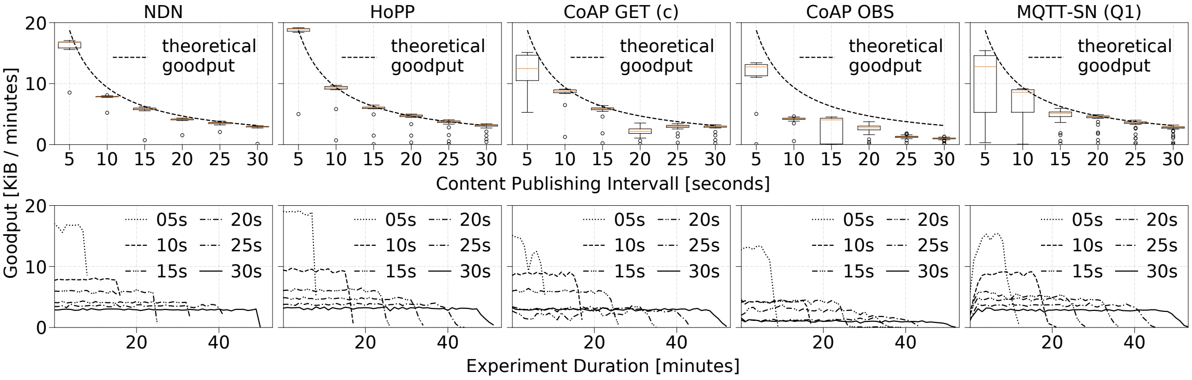


Figure 3.7: Goodput summary and flow evolution for all protocols at different publish intervals.

and the lack of retransmissions. MQTT-SN presents another extreme: the excessive amount of corrective actions, especially at the gateway—see the elevated maximum in Table 3.2—leads to an average expense that is fourfold. CoAP GET situates between both configurations with an average of ≈152 mJ. NDN operates reliably throughout the experiment (see Figure 3.4b) with a minimal number of packets in the network. Shortened retransmission paths with on-path caching are the key protocol features of NDN to reduce overall energy expenditures down to an average of ≈99 mJ and still maintain distinct success rates. Since HoPP counts a link-local signaling overhead for each published data, the total power consumption slightly elevate.

Last, we dive deeper into the flow balance of the different protocols and evaluate its effective data goodputs during various content publishing experiments. Figure 3.7 summarizes the results. We display the distribution of goodput from the different experiments in box plots and compare to the theoretical optimum (lines). Time series of data goodput further reveal the flow behavior as displayed in the lower row of the figure.

Clearly, HoPP exhibits the most evenly balanced flows and shows nearly optimal goodput values, closely followed by NDN. All other flow performances fluctuate with some tendency of instability when approaching its full transmission speed. Some IP-based flows in MQTT-SN and CoAP drop to lower delivery rates which is primarily caused by slow repeated end-to-end retransmission. Multi-hop retransmissions in this error-prone regime tend to cause additional interferences and accumulate transmission errors. As a consequence, protocols operate at reduced efficiency – for CoAP OBS protocol performance drops down to 50 %. The overall results show that the absence of flow control as in UDP/IP–based protocols make the protocols fragile. Hop-wise retransmission management as utilized in NDN and HoPP re-balances flows and explicitly demonstrates its benefits for the IIoT instead.

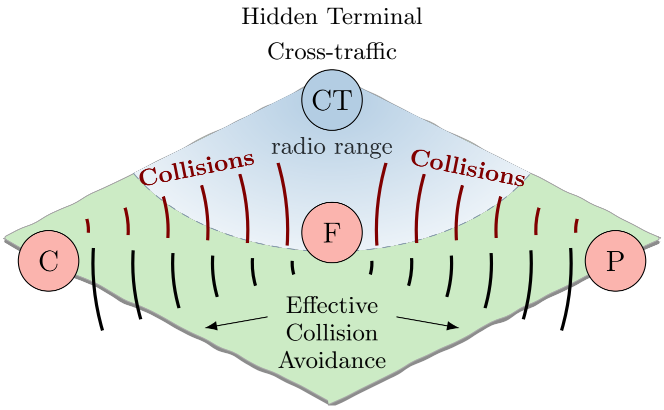


Figure 3.8: Experiment setup for measuring protocol resilience under cross-traffic.

3.5 The Impact of Coexisting Wireless Nodes

We continue our protocol analysis by investigating the case of uncontrolled disturbances. In unshielded environments, a frequent source of wireless degradation is caused by uncoordinated concurrent networks or by radiating appliances that interfere in the utilized frequency range. Such alien sources of disturbance are emulated by cross-traffic from a hidden terminal in our experiments.

3.5.1 Setup of Cross-traffic at Intermediate Hop

We examine the robustness of the networking protocols under cross-traffic using a two-hop topology between a content producer (P) and a consumer (C). Cross-traffic is injected towards an intermediate forwarder (F) as illustrated in Figure 3.8. We center (C) and (P) at (F) and verify in preceding measurements that both links perform comparably in both directions. By ensuring symmetry, we prevent a measurement bias with respect to antisymmetric sequencing of the different protocols. (CT) is our cross-traffic generator and placed next to (F), so that (F) overhears all transmissions, while (CT) remains hidden for (C) and (P). Hence, CSMA/CA fails for (C) and (P) and we expect an increased packet loss due to collisions for these nodes.

To scale the effect of cross-traffic at different stress levels, we configure the traffic generator in two dimensions as illustrated in Figure 3.9.

Burst Size reflects the number of consecutive packets sent to a third party link-layer unicast address. Each burst consists of a series of packets with a payload of 100 bytes.

Inter Burst Time denotes the pause in which our cross-traffic generator keeps the radio silent.

With varying cross-traffic patterns in place, we measure the error rates, data load, and time to completion for each protocol. We apply the periodic traffic pattern of one data unit every 1 s advised by our use case. Our first measurement validates the experimental environment.

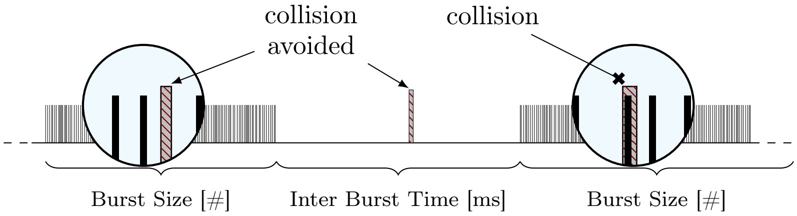


Figure 3.9: *Burst size* and *inter burst time* for our generated cross-traffic.

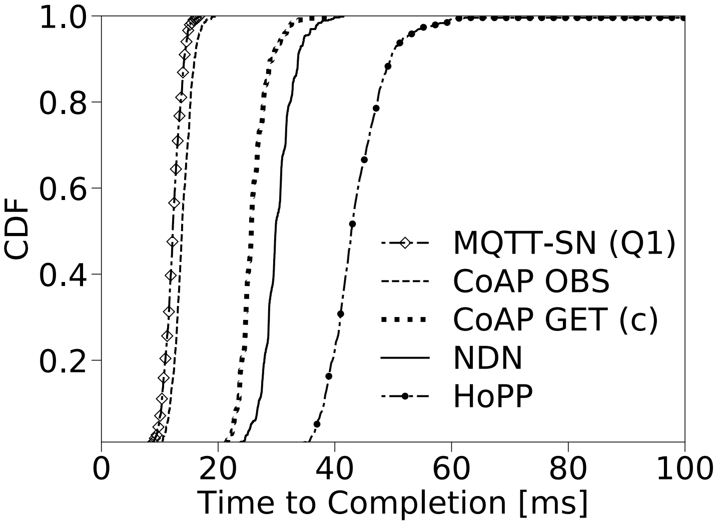


Figure 3.10: Time to content arrival in a two-hop topology at 1 s interval without cross-traffic.

Figure 3.10 displays the times to content arrival in the absence of cross traffic. All protocols perform as expected.

3.5.2 Results

Turning on the cross-traffic generator changes the picture. Figure 3.11 presents an overview of the protocol behaviors under 25 different scenarios of competing traffic. The color in each block visualizes the relative packet loss, while the numbers denote the relative redundancy of data packets on the links. A regular, undisturbed data packet traverses each link only once. Numbers higher than 1 indicate duplicate data packets, lower numbers indicate loss on the paths of data or request messages.

Decreasing the pauses between increasing bursts pushes the performance of all protocols below 50 % success rate. Still, the results are quite diverse. While the request-response protocols quickly degrade to error rates above 80 %, those protocols that push data (MQTT-SN and CoAP OBS) show a much higher chance of successfully transmitting data. It should be noted here that the CoAP OBS is unreliable and does not repeat data. Hence, its success rate turns

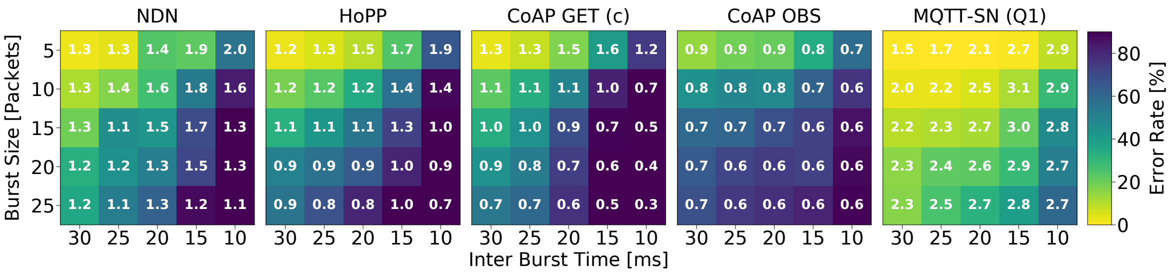


Figure 3.11: Error rate vs. data redundancy for a 1 s publishing interval. Colors encode errors and numbers tell the effective ratio of data packets sent over uniquely published items.

lower than MQTT-SN, while its data rate on the air also drops. On average, only ≈ 60 % of the data packets traverse both links, many of which only make the first hop. In contrast, MQTT-SN pushes packets via UDP until an acknowledgment arrives. This leads to a very high redundancy, which almost triples the data rates on the links. By pushing data intensely, though, MQTT-SN manages to attain superior performance among all protocols.

CoAP GET and the ICN protocols transmit data only on request. Since the cross-traffic jamming repeatedly destroys these requests, data is often not even transmitted. In consequence, data only sparsely appears on links even though these reliable protocols retransmit. The results are slightly better for the ICN protocols, since they transfer packets hop-wise with caching in place at the forwarding node.

This harsh, highly disruptive experimental regime reveals a significant heterogeneity among the protocols and their ability to co-exist on stressed links. While MQTT-SN accesses wireless resources rather aggressively – possibly on the expense of concurrent communication, the request-response protocols politely retreat from flooding data onto the congested link. It is noteworthy that data arrives in about equal shares among the four protocol retransmissions, so that about 25 % reaches the consumer only after 8 s. Reducing the retransmission timeouts would further increase the link utilization and lower the chances of a successful data transfer.

3.6 Related Work on Protocol Evaluation

3.6.1 Data dissemination in the Industrial IoT

Wireless communication plays an important role for connecting sensors and actuators in the IIoT and its heterogeneous systems. We have discussed current wireless link layers in Section 3.2, which are all error prone in the often harsh industrial deployments. Networking protocols on its upper layers may procure for high reliability as well as security needed for application scenarios such as control loops or safety related alerting. Challenges and requirements for typical IIoT scenarios have been investigated in [70]. Bernieri *et al.* [71] monitor factory automation systems and identify traffic anomalies in a hybrid system of traditional Modbus/TCP [72] as well as CoAP communication. Experimental evaluations of a distributed IoT data plane were recently presented in [73] and [74]. While the first work aims at optimizing the overall network throughout on edge nodes, the second introduces a lightweight messaging middleware to minimize resource consumption on low-end devices for edge computing.

Eggert [75] demonstrates on real IoT hardware the feasibility of using QUIC [76] for constrained devices. As a transport based on UDP, it provides a lightweight replacement for TCP with flow-controlled and multiplexed streams, a low-latency connection establishment, and built-in security features, which are valuable additions for safety-critical infrastructures. Extensions, such as Multipath-QUIC [77] and QUIC-FEC [78], bring an improved resiliency to connectivity failures. While comparative evaluations [79, 80] were mainly conducted for general purpose hardware, they yield promising results for deployments using multiple interfaces with high loss probabilities. Multiple interfaces on the same network hierarchy, however, are uncommon in industrial IoT deployments.

3.6.2 Performance evaluation of CoAP and MQTT

The performance of CoAP and MQTT have been studied from several perspectives over the last years [81, 82, 83, 84]. Very early work analyzed the interoperability of specific CoAP implementations [85, 86] without performance evaluation. Later, CoAP implementations have been assessed in comparison to HTTP [87] or on different hardware architectures [88]. MQTT was evaluated in [89] and compared to HTTP in [90]. Rodríguez *et al.* [91] analyze MQTT and HTTP using TCP/IPv4 as a transport. Thangavel *et al.* [92] proposed a common middleware to abstract from CoAP and MQTT. Based on this middleware, CoAP and MQTT were evaluated in a single-hop wired setup. In emulation, MQTT and CoAP have been studied in the context of medical application scenario [93]. Experimental analyses of MQTT and CoAP running on a hardware simulator (Cooja) have been presented by Martí *et al.* [94], and Proos *et al.* [95] perform measurements on a Raspberry Pi. The authors in [96] evaluate implications of the radio technology on higher layers protocols. They focus on the cellular 4G technology Narrowband IoT (NB-IoT). Still, a holistic analysis of these protocols in a consistent experimental setting including many real low-end devices with low-power wireless short range radio technologies is missing.

3.6.3 ICN and the IoT

The benefits of ICN/NDN in the IoT have been analyzed mainly from three angles. (*i*) design aspects [97, 98], (*ii*) architecture work [99, 100, 101], and (*iii*) use cases [102, 103, 104]. By stacking CoAP on ICN, Islam *et al.* [100] introduced CoAP as a convergence layer for applications that can run over both networking protocol stacks. Another approach [105] constructs a pure CoAP deployment option that replicates information-centric properties to gain the beneficial effects of ICN and still sustain protocol compliance with the CoAP specification [4]. Experimental evaluations are supported by several implementations that have become publicly available, including CCN-lite [106] on RIOT [28] and on Contiki [107], and NDN on RIOT [108].

The evaluation of NDN protocol properties in the wild includes the exploitation of NDN communication patterns to improve wireless resource management [109] as well as data delivery on the network layer [110, 39], which are to a larger extent reproducible with data-centric CoAP deployments [105].

We performed a first comparison with common IoT network stacks in [111]. This chapter extends our previous work and deepens the analyses in the context of the IIoT.

3.7 Conclusions

This chapter discussed and analyzed current networking solutions for the constrained Industrial Internet of Things. Starting from the challenging use case of safety-critical sensors and industrial control systems, we derived key requirements for the protocol behavior in a target deployment. Facing these requirements, we deployed and evaluated the three protocol families MQTT-SN, CoAP, and ICN in real-world experiments with settings characteristic for the IIoT.

Our analysis revealed that the choice of protocol largely impacts the application performance. On the overall, lean and simple publish-subscribe protocols such as MQTT-SN and CoAP Observe are versatile and operate efficiently in relaxed environments with low error rates. Request-response schemes hardly meet latency constraints of unscheduled alerts. Even though reliable, MQTT-SN and CoAP quickly fail in massive multi-hop scenarios, in which NDN and NDN-HoPP can both enfold strength of hop-wise transfer and reliably deliver data without the need for significant retransmission rates. MQTT-SN best withstands degradation from cross-traffic of coexisting wireless users—at the price of straining the overall resources by bursty (re-)transmissions. With these results, we hope to shed light on the role of networking and to strengthen deployment in the constrained IIoT.

Chapter 4

Low-Power Link-Layer Convergence for ICN

Abstract

The low-power Internet of Things (IoT) introduces lossy radio links with ultra-constrained frame sizes and high transmission cost for each byte. Information Centric Networking (ICN) is considered a promising communication technology in this regime, as it increases reliability by ubiquitous caching and eases transmission efforts by hop-wise forwarding. Common ICN layers such as NDN, however, were designed for fixed network infrastructure and require an adaptation layer to the constrained wireless—as the common Internet Protocol does.

In this chapter, we design and evaluate such an ICN convergence layer for low power lossy links that (*i*) augments the NDN stateful forwarding plane with a highly efficient name eliding, (*ii*) devises stateless compression schemes for standard NDN use cases with utile data encodings, (*iii*) adapts NDN packets to the small MTU size of IEEE 802.15.4, and (*iv*) generates compatibility with 6LoWPAN so that IPv6 and NDN can coexist on the same LoWPAN links. Our findings indicate that stateful compression can reduce the size of NDN data packets by more than 70 % in realistic examples, while packet fragmentation operates in a predictable way even for high fragment numbers. Our experiments show that for common use cases ICNLoWPAN saves 33 % of transmission resources over NDN, and about 20 % over 6LoWPAN.

4.1 Problem Space and Related Work

The Internet of Things inherently connects numerous devices of substantial heterogeneity. In this work, we focus on deployment use cases that bundle low-end and battery-operated microprocessors in wireless networks, where packet transmission distinctly dominates power consumption. The challenges we face in such scenarios are manifold and range from limited MTUs, lossy links and mobility to link layers that lack basic protocol features, such as multiple frame encapsulation formats (*c.f.,* EtherTypes in Ethernet).

NDN couples name-based routing from TRIAD [112] with stateful forwarding from DONA [15] and seamlessly leverages in-network caching on the forwarding plane. Three deployment scenarios for NDN are currently envisioned: (*i*) the leanest deployment runs NDN directly on top of a link layer, which requires a mapping of names to MAC addresses [109], (*ii*) NDN as an IP overlay allows to operate in the existing Internet infrastructures, and (*iii*) an integration of NDN encodings into IPv6 headers (see hICN [23]) can display synergies from the information-centric and the host-centric world. There is an additional inverse proposal that runs an IP overlay on top of NDN to continue the use of established, IP-bound applications and services, while taking advantage of loosely coupled content replication and in-network caching [113].

The fundamental request-response semantic on the network layer of NDN requires an *Interest* message and a returning *data* message. Both message types utilize flexible Type-Length-Value (TLV) header fields to allow for generic and extensible packet formats to the cost of space efficiency. Name TLVs are essential to NDN and thus always appear in Interest as well as in data messages. Depending on the naming scheme, human-readable Name TLVs make up the largest part of a request, and also of a response for many IoT use cases. We explore related work that copes with strict message length limitations using header compression and fragmentation in IP networks first, and then discuss proposed solutions for NDN.

IPv6 mandates a minimum MTU of 1280 bytes for each link and thus precludes a proper IPv6 operation in low power networks with small-sized MTUs. The IETF designed and extends a set of protocols for constrained IoT deployments where the 6LoWPAN convergence layer is an integral part of. It is situated below the network layer and provides packet encapsulation, stateless and stateful header compression as well as a protocol independent link fragmentation scheme. A generic header compression (GHC) [67] extends 6LoWPAN with an LZ77 flavored approach to deal with headers and header-like payloads that are not covered by the 6LoWPAN compression specification. While 6LoWPAN provides necessary elements for an interoperable and interconnected host-centric IoT, the same challenges remain open for information-centric IoT deployments.

Shang *et al.* [108] proposed a lightweight link fragmentation scheme that prepends a 3-byte fragmentation header to each NDN fragment to allow for messages larger than the limiting MTU of IEEE 802.15.4. This custom header further supports a minimal protocol identification by distinguishing normal NDN messages from fragmented messages. However, this protocol encapsulation collides with the 6LoWPAN encapsulation and thus disregards interoperability with 6LoWPAN, especially in multi-interface and multi-protocol deployments.

Another approach was presented by Mosko *et al.* [114], which extends NDN with a new message type that encapsulates each message fragment. It also adds complexity to the state machine for each peer to initialize fragment sequence numbers and perform corrective actions in case of drifting sequences after packet loss. A similar approach is the NDN Link Protocol (NDNLP) [115], which features fragmentation and reassembly as well as ARQ mechanisms. It provides packet encapsulation to distinguish between normal NDN messages and link acknowledgments. The last two approaches add overhead in terms of memory consumption and error-control messages which is a disadvantage for low power use cases.

A secure fragmentation for content-centric networks that does not rely on hop-by-hop reassembly and therefore decreases latency was derived by Ghali *et al.* [116]. Each fragment is securely signed according to NDN semantics and can be cached on intermediate routers. The authors propose a new ContentFragment message type that includes a Name TLV for forwarding purposes. Since NDN names are theoretically of unlimited length, duplicating names for each ContentFragment message adds a significant overhead, which naturally is controversial in constrained IoT networks. Individual fragment forwarding in the IP world has also shown several unwanted effects, which degrade network performance [117]. Furthermore, sporadic disruptions and mobility in LLNs do not guarantee a successful handover of each individual fragment to complete the reassembly.

Yang *et al.* [118] focused on bandwidth reduction and an improved storage utilization by translating long names into short names for local communication. In this regard, a sensor node registers a prefix at a sink node and receives a shorter prefix, *e.g.,* a hash as a name replacement. Ingress messages that traverse the sink node are updated to include the short name and egress messages respectively are updated to include the long name. The proposed local name translation is transparent to nodes outside the local IoT network. We argue nevertheless that a centralized approach to handle name translations with registration and cancellation procedures adds complexity, which limits the scalability in large deployments.

In the following section, we concentrate on a fully distributed convergence layer that preserves compatibility with IP LoWPAN, but takes advantage of the information-centric characteristics to obtain a generically applicable, efficient though uncomplex compressive encoding named ICNLoWPAN.

4.2 ICNLoWPAN

ICNLoWPAN provides a convergence layer that maps ICN packets onto constrained link layer technologies to enable pure NDN deployments without running as an overlay on top of IP. This convergence layer includes features such as link fragmentation, protocol separation on the link layer level as well as stateless and stateful header compression mechanisms. Figure 4.1 shows the overall network stack of a 6LoWPAN and an ICNLoWPAN deployment in parallel for a consumer, a forwarder, and a producer. Both convergence layers are situated between the link layer and the actual network layer, such that each message traverses them. This allows for a transparent operation without the need for modifications at the network layer.

4.2.1 6LoWPAN Dispatching Framework

6LoWPAN defines a dispatching framework [3], where each frame is prepended with a 1-byte dispatch type and a possible dispatch header. Several dispatch types exist already, e.g., for stateless IPv6 compression, a mesh header for mesh-under routing purposes, link fragmentation, and extensions to expand the limited universe of possible dispatch types. One of those extensions is the 1-byte page switch dispatch [119], which arranges dispatch types into 16 pages and signals

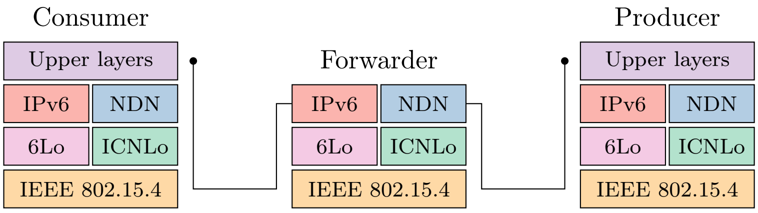


Figure 4.1: Stack traversal in a 6LoWPAN and ICNLoWPAN.

802.15.4 Header	RFC4944 Dispatches	Page 2 Dispatch	ICNLoWPAN Dispatches	802.15.4 FCS

Figure 4.2: IEEE 802.15.4 encapsulated ICNLoWPAN message.

a context switch to the packet parser to choose the proper page before interpreting subsequent dispatch types.

ICNLoWPAN integrates into the 6LoWPAN dispatching framework by defining four new dispatch types for Interest and data messages that are either compressed or uncompressed. Since page 0 and page 1 are already reserved for 6LoWPAN usage, we allocate these dispatches from page 2 to allow for coexistence with 6LoWPAN deployments. A prepended page switch dispatch before the very first ICNLoWPAN dispatch is thus necessary as an indicator to the dispatch parser.

A typical ICNLoWPAN message encapsulated in an IEEE 802.15.4 frame is shown in Figure 4.2. RFC4944 dispatches are optional and may include all dispatch types defined in [3]. Note the 1-byte page 2 dispatch before the first ICNLoWPAN dispatch. To switch back to 6LoWPAN dispatches after an ICNLoWPAN dispatch, another page switch dispatch to page 0 or 1 is necessary.

A major benefit of reutilizing the 6LoWPAN dispatching framework is to share a common code base for dispatch handling. Notably for multi-interface and multi-protocol deployments that use IPv6 and NDN simultaneously, having the same code components in resource-constrained devices is exceptionally valuable for minimizing RAM and ROM requirements.

4.2.2 Fragmentation

Reusing the 6LoWPAN dispatching framework enables ICNLoWPAN to seamlessly benefit from the protocol independent link fragmentation scheme defined in [3]. It is thus possible to fragment large NDN messages to fit the limited maximum physical packet sizes of low power link layers, such as 127 bytes for IEEE 802.15.4.

Practically, a fragmented NDN message includes a 4-byte fragmentation dispatch header that lists the original packet size and a packet tag to identify fragments of differing packets. Subsequent fragments further include an additional 1-byte datagram offset of the payload in the dispatch header. Fragments are reassembled on the next hop and passed to the NDN network

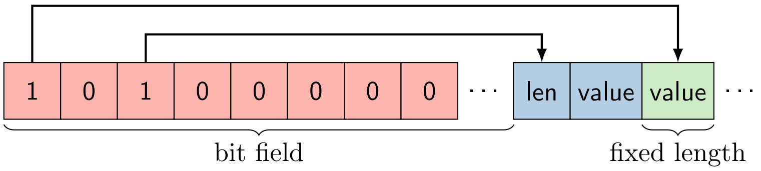


Figure 4.3: Eliding *type* and *length* fields using compact bit fields.

stack as typical NDN Interest or data messages. The 6LoWPAN fragmentation scheme does not define ARQ mechanisms to recover lost fragments, but rather relies on corrective actions of the link layer. This allows for implementations with minimal memory footprints.

6LoWPAN Fragment Forwarding (6FF) [120] and Selective Fragment Recovery (SFR) [121] are upcoming mechanisms that replace the original fragmentation scheme [3] with refined approaches. Instead of a hop-wise reassembly, 6FF forwards single fragments between endpoints. SFR further adds congestion control capabilities and enables a selective recovery of lost fragments, which has been extensively evaluated in an IoT testbed [117, 122]. These mechanisms open up new possibilities for further optimizations of the ICNLoWPAN integration [123].

4.2.3 Stateless Compression

ICNLoWPAN defines a stateless header compression scheme with the main purpose of reducing header overhead of NDN packets. This is of particular importance for link layers with small MTUs and for increasing energy conservation of battery-operated and wirelessly connected devices. Corresponding dispatch headers in the ICNLoWPAN packet provide the rule set for decompressing NDN messages before handing the packet over to the NDN network stack.

TLV Compression

The NDN header format is solely composed of TLV fields to encode header data. The advantage of TLVs is a native support of variable-sized data. The main disadvantage of TLVs is the verbosity that results in two extra bytes for each header field to store the *type* and *length* of the encoded data.

The stateless header compression scheme of ICNLoWPAN makes use of compact bit fields to indicate the presence of optional TLVs in the uncompressed packet as illustrated in Figure 4.3. Each *type* that is present in the bit field is thus elided from the actual TLV representation, which translates to a reduction from 1 byte to 1 bit. Further compression is achieved with eliding the *length* of TLVs that either represent fixed-length header data, or where the length can be assumed from surrounding TLVs. We also achieve smaller encodings by specifying sane default configurations for IoT use cases.

Name Compression

A Name TLV is substantial to NDN messages and usually consists of several name components, each of variable size. An Interest message essentially carries the Name TLV and a data message returns either the same TLV, or a more specific variant with an equal prefix. The NDN TLV encoding requires at least two bytes for each name component (*type* + *length*) and an extra two bytes for the outer-most Name TLV. The TLV overhead for a name is displayed in Eq. 4.1, where $|c|$ is the number of components.

$$TLV\ overhead = 2 + 2 \cdot |c| \tag{4.1}$$

ICNLoWPAN provides a compression scheme for Name TLVs that drastically reduces the TLV overhead of each nested component. This compression encodes length fields of two consecutive component TLVs into one byte, using 4 bits each as displayed in Figure 4.4. This process limits the length of a component TLV to 15 bytes. To further elide the outer-most *length* field of the name, this scheme utilizes a stop marker. For an odd number of components, the stop marker is encoded into the least significant 4 bits of the current *length* byte. On an even number of components, the full *length* byte is already occupied with two name components. In this case, a stop byte is appended to the last component as shown in Figure 4.4.

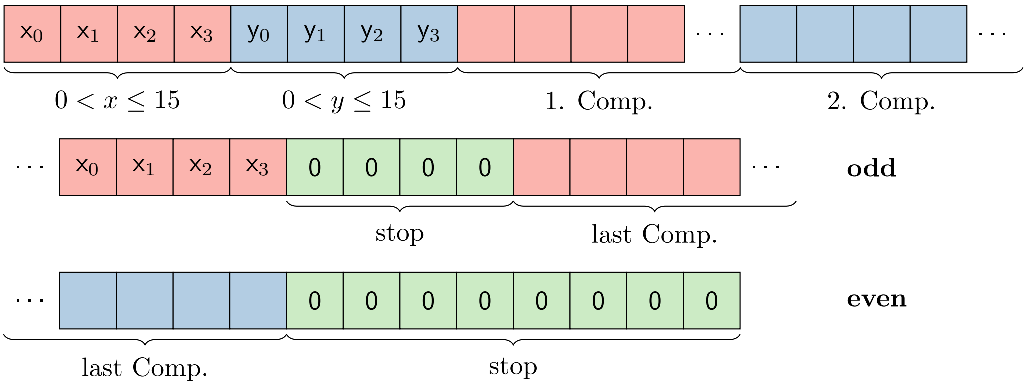


Figure 4.4: Stateless name compression and stop marker for odd and even number of name components.

Compressed names yield a significantly lower TLV overhead as displayed in Eq. 4.2, where $|c|$ again is the number of components. The ceil operator handles both cases of odd and even for $|c|$.

$$TLV\ overhead = \left\lceil \frac{|c| + 1}{2} \right\rceil \tag{4.2}$$

The total TLV overhead reduction for names thus follows from Eq. 4.1 and Eq. 4.2 and is a function of $|c|$. The reduction is therefore given as $\lceil 1.5 \cdot |c| \rceil + 1$.

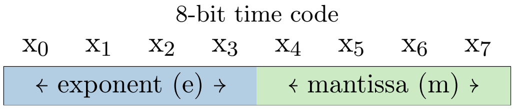
Efficient Encoding of Timestamps

NDN utilizes timestamps for various protocol operations and thereby differentiates between absolute and relative values. Relative timestamps denote positive time offsets that add to the absolute message reception time and provide a general sense of recency. Absolute time values indicate the elapsed time since the Unix epoch and grant a higher precision for global time and date values throughout the network.

The current NDN packet format describes three TLVs that encode temporal values. Interest messages optionally carry `InterestLifetime` in relative time. Data messages optionally include `ContentFreshness` in relative time and the `SignatureTime` as an absolute timestamp. All timestamps in respective packet headers are expressed as milliseconds.

At the time of writing, the current absolute time in milliseconds is encoded in 40 bits. Absolute timestamps in TLV form will thus use 8 bytes to denote current or future moments in time. Relative time offsets on the other hand encode in fewer bytes and still provide an effective protocol operation: A 2-byte value represents a maximum offset of around 65 seconds and a 4-byte value holds a maximum of around 49 days. In typical IoT use cases with long sleep cycles and disrupted network connectivity, the 4-byte form is much more likely to be utilized for the lifetime of Interests and content freshness.

ICNLoWPAN uses a compressed representation for time offsets, *i.e.,* `Interestlifetime` and `ContentFreshness`, similarly to previous work based on CCNx [124]. Instead of a linear representation, ICNLoWPAN encodes time values in dynamic range to limit the size of offsets to a single byte. This form allows for high precision between lower time values and at the same time enables wide time ranges. Following the standard for floating-point arithmetic [125], an 8-bit time code describes the values for exponent and mantissa as shown in the following equations. A configurable number of high-order bits denotes the exponent value and the remaining low-order bits indicate the mantissa value. Eq. 4.3 shows a subnormal form inspired from IEEE 754 [125], which applies when the exponent value is 0. The subnormal form allows to represent 0 seconds and fills the underflow gap with time values to enable a gradual underflow. For all other cases, the normalized form in Eq. 4.4 is used to convert a time code to a time value in seconds. In both equations, $m_{\max}$ denotes the maximum mantissa value that a time code can represent and b is a bias that is statically configured.



$$\underset{\substack{\text{time value}\\ \text{in seconds}}}{t(\mathrm{e}, \mathrm{m}, b)} = \begin{cases} \left(0 + \dfrac{\mathrm{m}}{m_{\max}}\right) \cdot 2^{(1+b)}\, s & e = 0 \qquad (4.3) \\ \left(1 + \dfrac{\mathrm{m}}{m_{\max}}\right) \cdot 2^{(\mathrm{e}+b)}\, s & e > 0 \qquad (4.4) \end{cases}$$

Suitable sizes for exponent, mantissa, and bias determine the effectiveness and applicability of this efficient encoding of timestamps. Sec. 4.3.2 explores different configurations and outlines recommended values.

4.2.4 Stateful Compression

ICNLoWPAN further employs two stateful compression schemes to enhance size reductions. These mechanisms rely on shared contexts that are either distributed and maintained in the whole LoWPAN, or are generated and maintained on-demand for a particular Interest-data path. Our stateless and stateful compressions are applied in succession to produce huge compression savings, which we show in Sec. 4.3.

LoWPAN-local State

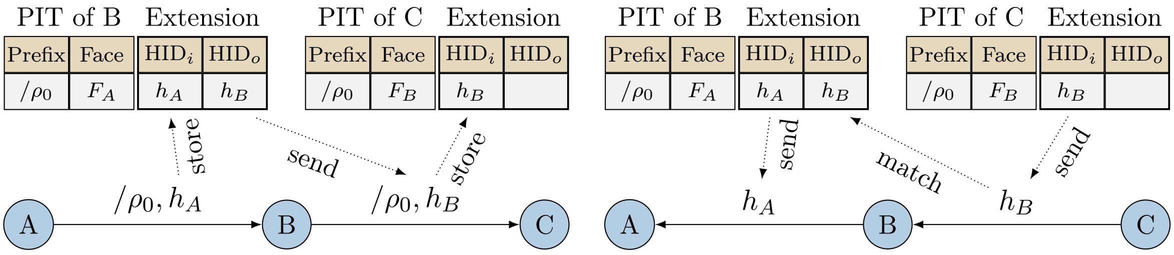
A context identifier (CID) is a 1-byte number that refers to a particular conceptual context between networked devices and may be used to replace frequently appearing information, like name prefixes, suffixes, or meta information, such as an Interest lifetime. This allows for a reduction of potentially long data to a single byte. Shared context has to be initially distributed at compile-time or dynamically maintained on run-time in order for a device to properly encode and decode NDN messages.

The convergence layer replaces header fields of outgoing Interest and data packets with CIDs maintained in a CID state table. Context identifiers follow the last ICNLoWPAN dispatch, while the most significant bit of a CID signals the presence of a subsequent CID. On reception, the original packet is restored and passed to the network layer.

En-route State

An NDN Interest requests data by a name or a prefix which is then returned in the corresponding data packet. This duplication generates large overhead in particular for long names. To deduplicate we make use of ephemeral 1-byte HopIDs that replace the name in data responses and link them to entries of the *Pending Interest Table (PIT).* The PIT is a fundamental component of NDN that hop-wise matches the returning responses to open requests by name. HopIDs must be unique within the local PIT and only exist during the lifetime of a PIT entry. We extend the PIT by two new fields to manage these HopIDs. HID_i for inbound HopIDs and HID_o for outbound HopIDs as visualized in Figure 4.5. We emphasize that even though PIT entries are extended by two additional bytes, the overall RAM consumption on link commitment reduces due to smaller packets and corresponding message buffers.

Before sending an Interest, a node generates a HopID and stores it in the local HID_o column. This Interest then includes the generated HopID along with the name. On the next hop, the HopID is extracted from the Interest and stored in the HID_i column of the respective PIT entry. The forwarder then generates a new HopID, stores it in the HID_o column of the particular PIT entry, and puts this HopID into the Interest message before it is forwarded to the next hop.



(a) Collecting compression state en-route (Interest). (b) Eliding Name TLVs using en-route state (data).

Figure 4.5: Stateful header compression using en-route forwarding state.

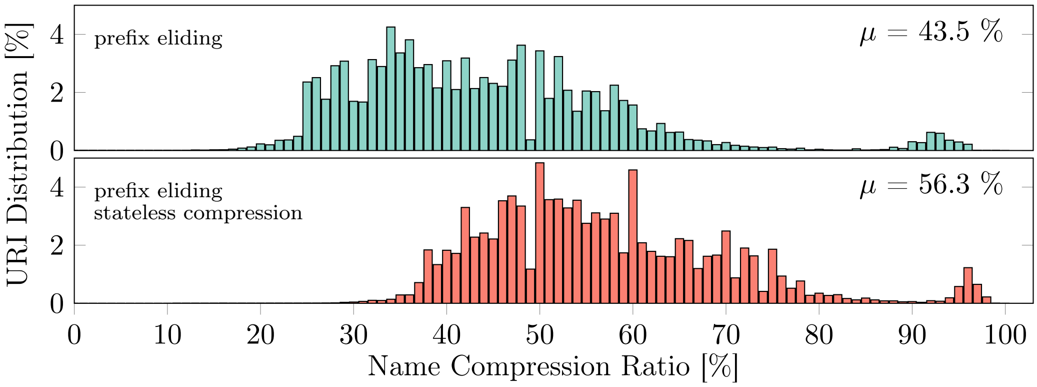


Figure 4.6: Distribution of percental name compression ratios.

This process is repeated for each hop until the request can be satisfied with the corresponding response as displayed in Figure 4.5a.

The producer of a returning data message reverses this process by obtaining a HopID from the HID_i column of a PIT entry and encodes it into the response message. If the returning name equals the original name, then it is fully elided. Otherwise, the distinct suffix is included along with the HopID. When a response is forwarded, the contained HopID is extracted and used to match against the correct PIT entry by performing a lookup on the HID_o column. The HopID is then replaced with the corresponding HopID from the HID_i column before forwarding the response, as visualized in Figure 4.5b.

4.2.5 Name Compression Performance

Our proposed stateless and stateful name compression mechanisms are of simple nature and do not exhaust scarcely available CPU resources. The compression ratio is highly dependent on (*i*) the number of name components as they dictate the overall TLV overhead and (*ii*) the length of the prefix omitted from the name. To quantify the effects, we analyze the average name compression performances in Figure 4.6 for $356 \cdot 10^6$ real-world URI paths obtained from the WWW, where each component does not exceed 15 bytes. Before applying our compression scheme, we encode all names as NDN Name TLVs. First, we elide the hostname from each URI as part of our stateful compression in Sec. 4.2.4. This yields an average compression ratio of 43.5 %. Second, we apply the stateful name compression described in Sec. 4.2.3 to reduce the TLV overhead and observe a compression ratio of 56.3 % on average. We believe that typical names in a low power IoT edge network will yield similar, if not better, compression performance.

4.3 Evaluation

4.3.1 Experiment Setup

Protocol Comparison

We consider two different NDN deployments for low power IoT networks. In the first experiment, we run NDN directly on top of IEEE 802.15.4. Due to the small MTU of this particular link layer, we limit packet sizes to a maximum of 100 bytes. In the second experiment, NDN runs on top of ICNLoWPAN with activated stateless and stateful compression. We further compare our NDN setups with a typical 6LoWPAN operation that uses UDP as a transport protocol and CoAP as an application protocol. We specifically compare against the GET method of CoAP as it provides a request-response pattern analog to NDN. We deploy our devices in two different network configurations.

Single-hop. A consumer device has managed FIB entries to a producer device and vice versa. In our NDN deployment, the producer initially creates all content objects. The 6LoWPAN producer configures a callback function as a CoAP endpoint for a particular URI to trigger a response.

Multi-hop. An extra forwarding device is added to the network topology, such that requests and responses between the consumer and producer traverse through the forwarder.

Hardware & Software Platform

We conduct all our experiments on typical class 2 [1] devices that feature an ARM Cortex-M0+ MCU with 32 kB RAM, 256 kB ROM and up to 48 MHz CPU frequency. Each device further provides an Atmel AT86RF233 [69] 2,4 GHz IEEE 802.15.4 radio transceiver. We set the radio transmission power to 0 dBm, the receiver sensitivity to -94 dBm and enable the *Smart Receiving* feature, an energy saving mode for idle listening. For our power consumption measurements we make use of on-board current measurement headers on each of our devices. We measure currents using a Keithley DMM7510 $7\frac{1}{2}$ digit graphical sampling multimeter with 1 MHz sampling rate and control it with external I/O lines to trigger start and stop from events generated by our network stack. Devices under test are powered by a regulated external DC power supply and connect via UART to a Linux control node to obtain experiment results.

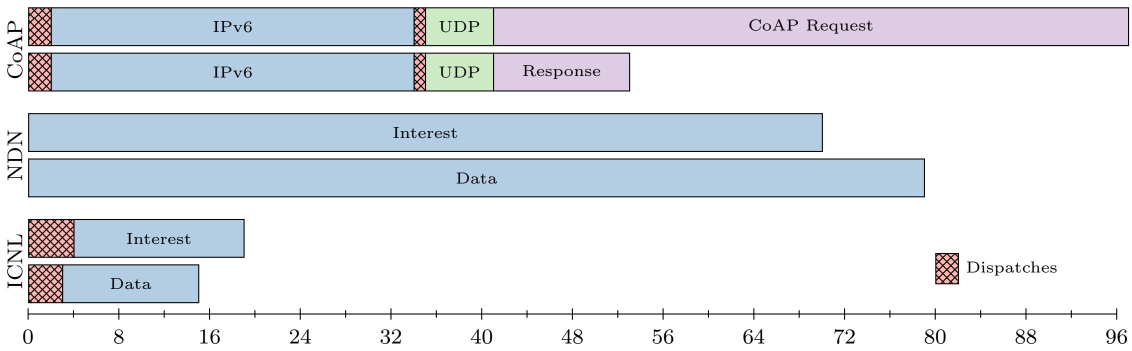


Figure 4.7: Packet length and structure for different protocols.

In each experiment, our devices operate RIOT OS version 2018.10. Our NDN deployments use the CCN-lite [106] package and our 6LoWPAN experiments are based on the default GNRC network stack of RIOT OS. We integrate ICNLoWPAN into the 6LoWPAN module of RIOT OS to reutilize the code base of the dispatching framework and link fragmentation.

Name Configuration

Name lengths proportionally affect processing times, packet lengths and consequently energy expenditure during transmissions. This especially impacts NDN as names are included in requests as well as in responses. We use two different names in our experiments to measure the effects of our stateless and stateful compression mechanisms.

Name$_{short}$. We use a short name with 4 components to denote temperature readings produced by a sensor. The name is of the form `/org/example/temp/`id_x, where id_x is an increasing number for each request. A CID is configured for `/org`, such that this prefix is elided from all messages.

Name$_{long}$. We use a long name with 10 components of the form `/org/example/building/1/floor/4/room/481/temp/`id_x. A CID is configured for the prefix `/org/example/building/1/floor/4/room/481` to elide a considerable portion of the name.

4.3.2 Theoretical Evaluation

Packet Dissection

In this first evaluation, we analyze NDN and CoAP packet sizes for a typical IoT scenario using Name$_{long}$ as a CoAP endpoint and as a naming scheme for NDN. We further configure responses of both protocols to return 4-byte signed integer values that represent temperature sensor readings.

Figure 4.7 depicts the actual packet sizes for each protocol. Our CoAP request has a packet size of 97 bytes, where 3 bytes are used for 6LoWPAN dispatches, 32 bytes for the compressed IPv6 header and 6 bytes for the UDP header. The remaining 56 bytes are used by the actual

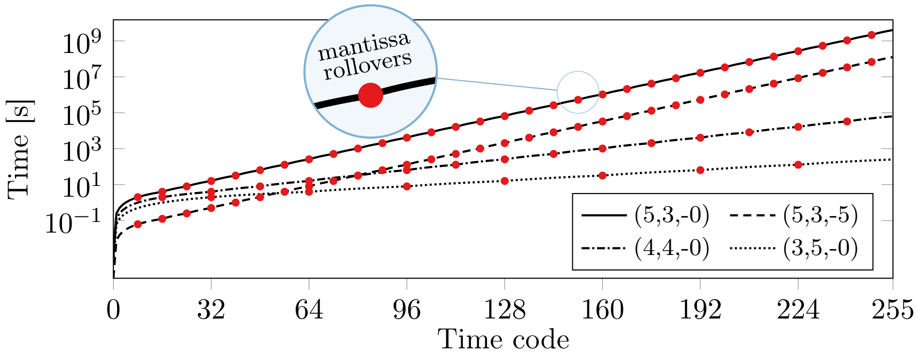


Figure 4.8: Precision and range of various timestamp encoding configurations.

CoAP message. The respective CoAP response requires considerably less, which follows from CoAP omitting URIs in responses and using tokens to match against open requests on the requesting node. In our setup, CoAP uses 2-byte tokens for each request and returns the exact token in the response. The NDN Interest message nets to 70 bytes, whereas the returning data message requires 79 bytes. Contrary to CoAP, returning responses in NDN include the name of the request, or even a more specific and longer name. The displayed data message further contains empty Signature TLVs. The equivalent ICNLoWPAN compressed NDN messages are significantly shorter, where the Interest message reduces down to 19 bytes (72 % savings) and the data message down to 15 bytes (81 % savings). For Interests, this gain is mainly due to leveraging the configured stateful name compression and data messages naturally benefit from eliding the full returning name. In addition to the compressed messages, we require a 1-byte page dispatch, 1-byte ICNLoWPAN dispatch, and a 1-byte HopID for Interest and data. The compressed Interest packet further includes a 1-byte CID indicating the elided prefix for the stateful name compression.

Configurations for Compressed Timestamps

In the next evaluation, we explore different configurations of the enhanced timestamp encoding described in Sec. 4.2.3. Figure 4.8 illustrates the precision and maximum ranges in logarithmic scale for selected configurations. The number of bits for exponent, mantissa, and the value for bias are noted in the form `(exponent,mantissa,bias)` for each plot. Red dots symbolize mantissa rollovers and indicate an exponent increase by one. The configuration (3,5,-0) allocates five mantissa bits and thus shows a precision loss that declines minimally across mantissa rollovers as indicated by the distances between the red circles. Conversely, the remaining three bits for the exponent allows only for a small maximum range of 252 seconds, while the minimum non-zero number is 62.5 ms. This configuration is unsuited for scenario deployments that expect a prolonged protocol operation (*e.g.,* request lifetime, content freshness) due to intermittent connectivity and network partitioning. Configuration (4,4,-0) increases the range to about 17 hours at the cost of less precision. The maximum time value boosts with configuration (5,3,-0): Time codes can represent time values as high as 127 years, but experience a considerable precision loss, even in the lower seconds range. 250 ms is the minimum non-zero time that can be represented. After 32 seconds, this configuration yields a 4-second resolution, which decreases to an 8-second resolution after 64 seconds. At around 10 minutes, the resolution is at 64 seconds. Configuration (5,3,-5) applies a bias of -5 to the previous setting and makes a good compromise by shifting the co-domain further into the sub-seconds range. It thereby sacrifices the maximum range and inherits the same precision decline from the latter configuration. The maximum representable time value degrades from 127 years to approximately four years. However, the negative bias adds a decent precision to the sub-seconds range, which is important in real-time scenario deployments that operate in milliseconds. This setting has a fine-grained resolution of 7.8 ms between 0–125 ms and 15.6 ms between 125–250 ms. The decent precision in lower sub-second ranges and the maximum range of four years allows this configuration to provide an effective operation for most of the IoT deployment scenarios.

4.3.3 Experimental Evaluation

The theoretical evaluation indicates that ICNLoWPAN substantially reduces packet sizes in Named-Data IoT networks with pull-driven traffic patterns. In this experimental evaluation, we want to explore the effects of our convergence layer on resource-constrained nodes to gauge the feasibility for real-world low power networks. In particular, we measure (*i*) intra-stack processing times, (*ii*) average message overhead for a request-response handshake, (*iii*) energy expenditure during transmissions for single-hop and multi-hop deployments, (*iv*) and effects on reliability when periodically requesting sensor values while we enable randomized and bursty cross-traffic.

Processing Times

We first evaluate intra-stack processing times in a multi-hop deployment, where a consumer requests every 500 ms a specific temperature reading from a producer using Name_{short} and Name_{long}. Timestamps for the link layer, convergence layer (LoWPAN), and upper layers are recorded for each packet using a hardware timer on each device with μs precision. The link layer time depicts operations of the RIOT OS radio driver including SPI transfer of the packet to the radio frame buffer with 5 MHz. This measurement does not contain the actual transmission or reception of a frame over the wireless medium, as this procedure does not load the CPU. Time spent in the LoWPAN module includes the handling of dispatch headers and packet (de-)compression. Processing times for the network layer and beyond either include IPv6, UDP, and CoAP or NDN operations in addition to the actual application on top that issues or satisfies requests.

Figure 4.9 displays experiment results for the different roles of a consumer, forwarder, and producer. We first observe that for the Name_{short} configuration, the additional processing

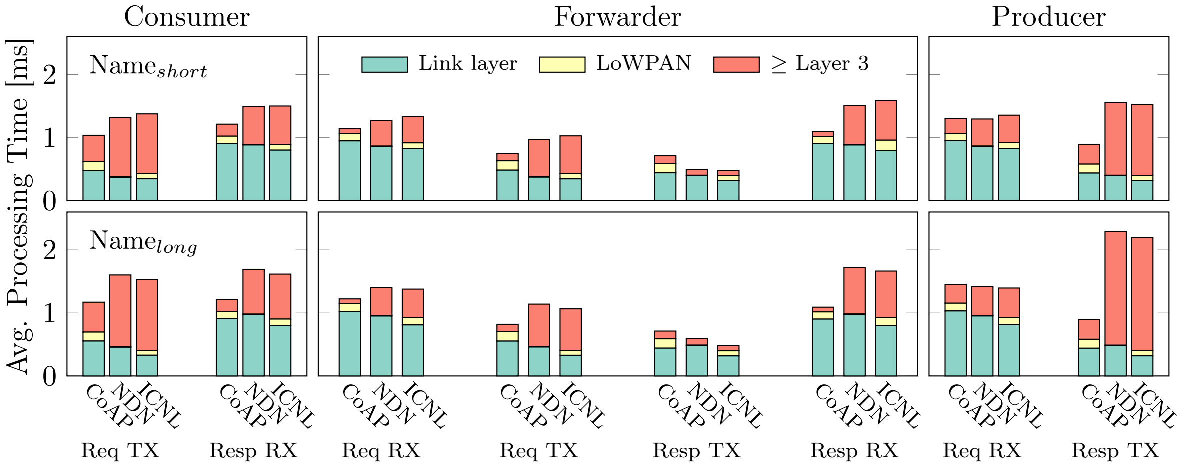


Figure 4.9: Intra-stack average processing times for CoAP, NDN, and ICNLoWPAN.

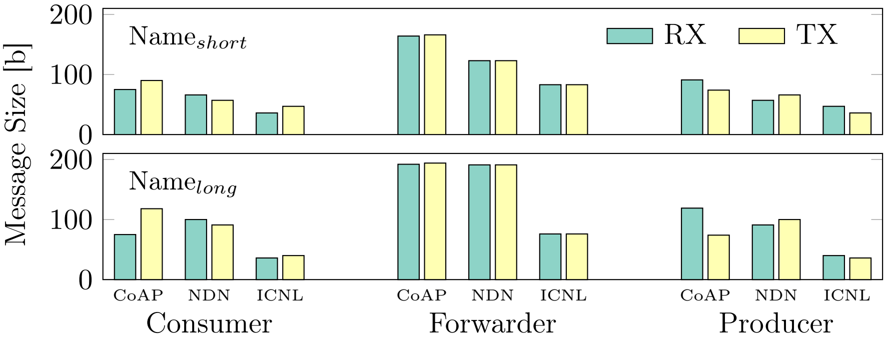


Figure 4.10: Average bytes per request—response.

overhead of ICNLoWPAN does not pay off on the link layer. Thus, some measurements with ICNLoWPAN take slightly more CPU resources than plain NDN. Conversely, savings on the link layer for our Name$_{long}$ configuration visibly outperform the ICNLoWPAN processing overhead, especially for response packets by a decrease of $\approx$ 100 μs per packet.

Message Sizes

We now analyze the amount of bytes that are transmitted between consumer, forwarder, and producer when performing a request-response handshake in Figure 4.10. The captured packets include the message lengths of Figure 4.7 in addition to 21 bytes for an IEEE 802.15.4 header and 2 bytes for FCS. While comparing the Name$_{short}$ and Name$_{long}$ configurations, we observe an increase in sent bytes for the CoAP and NDN consumer, whereas ICNLoWPAN reduces the amount of sent bytes due to a better utilization of our name compression scheme. We interestingly see that the amount of sent bytes for the CoAP producer stagnates, while the amount for our NDN producer increases. This follows from the fact that CoAP responses do

	Consumer		Forwarder		Producer	
	$Name_{short}$	$Name_{long}$	$Name_{short}$	$Name_{long}$	$Name_{short}$	$Name_{long}$
CoAP	548.58 μJ	612.24 μJ	967.41 μJ	1072.07 μJ	464.73 μJ	517.96 μJ
NDN	526.23 μJ	687.26 μJ	880.68 μJ	1152.02 μJ	422.55 μJ	584.82 μJ
ICNL	466.09 μJ	487.32 μJ	769.17 μJ	773.97 μJ	369.84 μJ	395.19 μJ

Table 4.1: Energy consumption in μJ.

not include the URI component, but rather a fixed-length token to match against open requests. In contrast, the NDN data message does include the full name that is obtained from the request, which leads to significantly increased message sizes from Name$_{short}$ to Name$_{long}$.

Energy Consumption

ICNLoWPAN decreases the total amount of bytes over the air with both name configurations compared to NDN. We observe a strong reduction of bytes for responses at the producer for both name configurations, since the name is fully elided. The reduction is most prominent for the Name$_{long}$ configuration at the forwarder with a drop by 60 % from 191 bytes down to 76 bytes. This gain is to be expected, considering that the forwarder is involved in four transmissions.

The packet length during transmission has an immediate effect on the energy consumption. We measure the current while sending and receiving messages for each role separately in a single-hop setup and display the results in Figure 4.11 for Name$_{long}$. The graphs involve transmission over the wireless, radio turnaround time as well as link layer frame acknowledgment. In our setup, sending draws slightly higher current than receiving and the duration of each transmission depends on the packet length. In fact, the duration of each depicted measurement correlates with the respective message size displayed in Figure 4.7 and the results showed in Figure 4.10, so that larger messages yield longer periods of operation for sending and receiving.

On the consumer, we observe that our CoAP request requires 5 ms to complete, while the respective NDN request is transmitted in 4 ms, including the reception of acknowledgments for both. Conversely, the CoAP response is received by the consumer in 3.8 ms, while the NDN response completes in 4.2 ms, including the sending of acknowledgments. With ICNLoWPAN in operation, we notice a decrease of transmission times by around $\approx$ 50 % on the consumer due to compressed messages and the resulting shortened media utilization. As expected, the reduction for responses is more prominent due to fully eliding Name TLVs. On the producer, we naturally observe mirrored results for each operation.

Given the fact that current draws for transmissions in Figure 4.11 are mainly similar, the actual energy consumption is dominated by the transmission durations. We thus analyze the overall power consumption of a request-response handshake in a multi-hop setup for Name$_{short}$ and Name$_{long}$ including processing times and transmission durations in Table 4.1. Our results

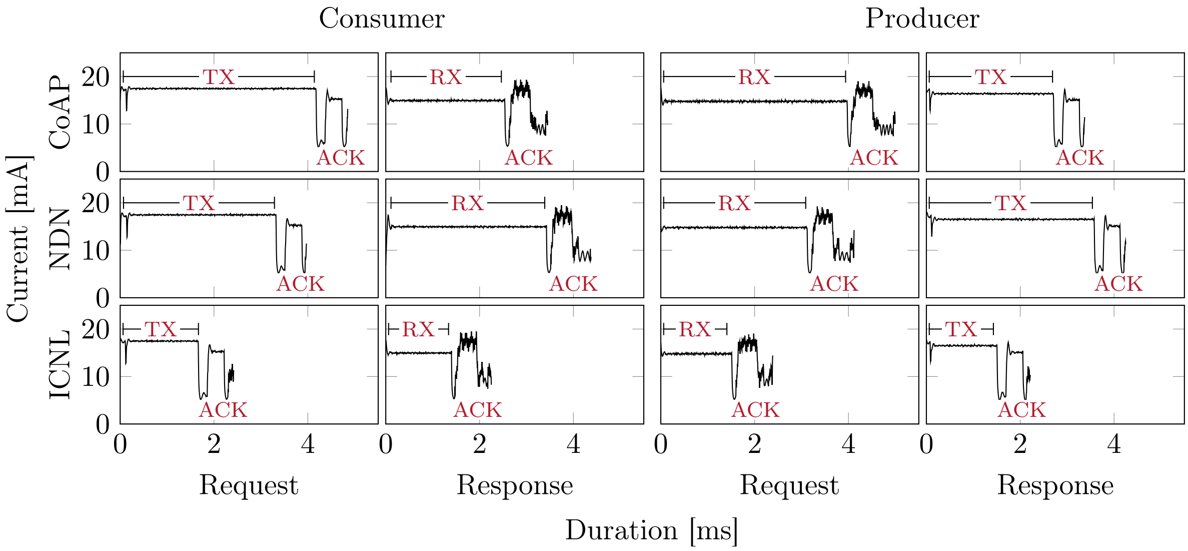


Figure 4.11: Current consumption for send and receive operations.

indicate an increase in energy usage for each role with the Name$_{long}$ configuration compared to the Name$_{short}$ configuration. We further notice that our producer spends the least amount of energy, followed by the consumer, and our forwarder expends nearly double the amount of energy than the producer. The increased power consumption is inherently consistent with the fact that the forwarder is involved in two request and subsequently in two response transmissions.

The NDN consumer device uses 4 % less energy for Name$_{short}$ and 12 % more energy for Name$_{long}$ compared to the CoAP consumer. This turnaround in energy expenditure for Name$_{long}$ is twofold. (*i*) NDN has a more verbose name encoding than CoAP and (*ii*) CoAP does not include the URI in the response. ICNLoWPAN reduces energy usage of our NDN consumer by 11 % for Name$_{short}$ and 29 % for Name$_{long}$. Our NDN producer consumes 9 % less energy for Name$_{short}$ and 13 % more energy for Name$_{long}$ compared to our CoAP producer. The energy consumption reduces by 12 % for Name$_{short}$ and 32 % for Name$_{long}$ with an enabled ICNLoWPAN operation compared to NDN. Since our forwarder interacts with four transmissions, we observe a natural increase in overall expenditures. The NDN forwarder consumes 9 % less power for Name$_{short}$ and 7 % more energy for Name$_{long}$ compared to the CoAP forwarder. In contrast, ICNLoWPAN reduces the expenditure by 13 % for Name$_{short}$ and 33 % for Name$_{long}$. The trend that ICNLoWPAN yields higher energy savings for Name$_{long}$ becomes apparent.

Finally, we calculate the energy consumption of a full request-response handshake for a multi-hop setup with a varying number of forwarders between a consumer and producer in Figure 4.12. For all setups, we see an increase in expenditure for CoAP, NDN, and ICNLoWPAN with an increasing number of forwarders. We again notice that the power consumption for NDN surpasses the consumption for CoAP using the Name$_{long}$ configuration due to returning Name TLVs in the response. ICNLoWPAN clearly reduces the overall energy consumption of an NDN

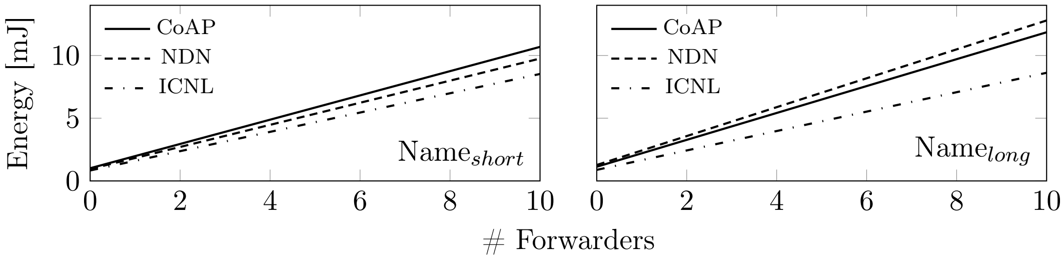


Figure 4.12: Total energy consumption for multi-hop networks.

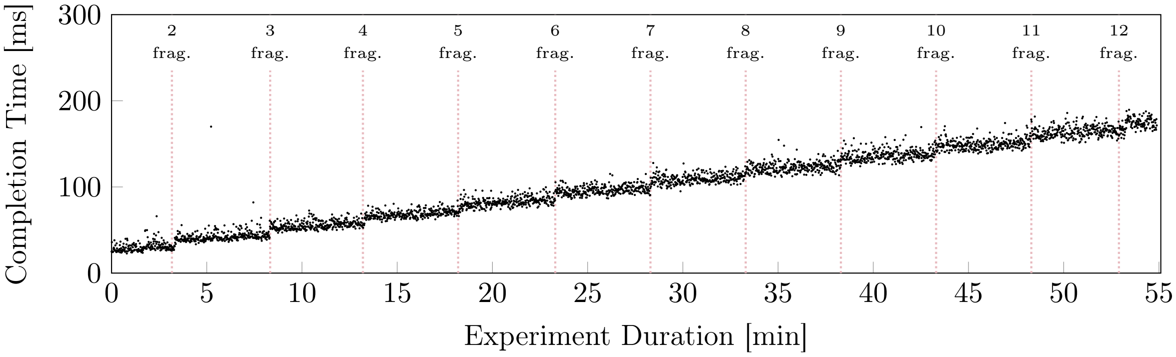


Figure 4.13: Content arrival times with gradually increasing payload sizes.

handshake for both configurations Name$_{short}$ and Name$_{long}$. Interestingly, despite the increase in power consumption for NDN versus CoAP, ICNLoWPAN manages to cut expenditures by ≈ 25 % for a setup with ten forwarders compared to NDN.

The Impact of Fragmentation

ICNLoWPAN employs a hop-wise fragmentation scheme for messages that do not fit the maximum physical packet size of 127 bytes in IEEE 802.15.4 environments. In this experiment, we gradually increase the payload size in data messages by 32 bytes after every 100$^{\text{th}}$ packet to examine the impact of fragmentation in our multi-hop scenario. We start with a payload size of 0 bytes and end at 1024 bytes. A consumer periodically requests content from every $1 \pm 0.1s$ using the Name$_{short}$ configuration. The requests traverse a forwarding node to reach the producer device.

Figure 4.13 portrays content arrival times at the consumer over the duration of the experiment. The gradual increase of returning payloads leads to elevated completion times as indicated by the previous experiment in Sec. 4.3.3. We observe a moderate incline of around 3 ms for payload sizes that do not provoke further fragmentation. As soon as sizes force an added fragment, the average completion time for subsequent content requests jumps by roughly 10 ms. We also note the increasing dispersion with each additional fragment due to accumulating link-layer activities, such as carrier sensing related back-offs and packet retransmissions. The scattering naturally

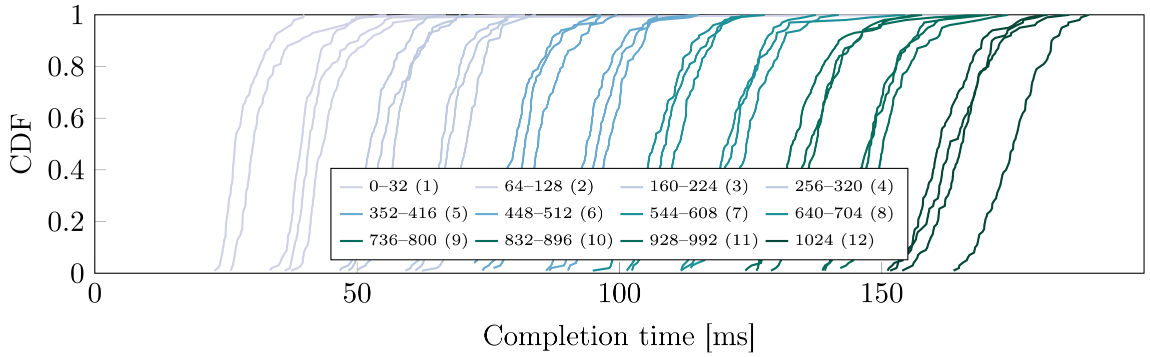


Figure 4.14: Temporal distributions of content arrival times for different payload sizes in bytes. The amount of fragments is given in parentheses.

shows its extreme spread for data messages with 11 and 12 fragments. Temporal distributions for content arrival times reflect in Figure 4.14 and confirm our previous results: Configurations with payload sizes that fit into the same amount of fragments are roughly 3 ms apart, while new fragments add another 10 ms to completion times. While 50% of packets with a payload size of 0–32 bytes finish below 25–30 ms, we observe an increase that is almost six-fold for payload sizes of 1024 bytes. This configuration generates 12 fragments per packet and 50% of requests complete within 175 ms.

We further want to quantify the processing overhead that results from fragmented messages. Figure 4.15 displays the average processing time spent within the fragmentation module for the previous experiment. Interest and Data packets that fit into a single fragment exhibit only a minimal processing overhead of 10–14 μs for transmission and reception on the consumer and producer side, respectively. The forwarder demonstrates a doubling due to processing Interest and Data messages in both directions. We observe a sudden increase in processing time for Data messages as soon as the fragmentation strategy activates: The producer shows an accumulated average processing time of 100 μs for fragmenting the data, while the consumer spends around 200 μs during the reassembly of two fragments. These numbers linearly increase with the number of fragments by approximately 70 μs for an added fragment and about 100 μs for the reassembly. For 12 fragments, the absolute numbers multiply up to 1.2 ms for the producer and forwarder on fragmentation and 0.7 ms on reassembly for the forwarder and consumer.

Reliability

In this experiment we investigate the reliability of a typical data retrieval setup in our single-hop deployment, where a consumer periodically requests temperature values from a producer every 300 ms. We additionally generate bursty cross-traffic in randomized intervals to a third party in order to mimic a dense network with multiple devices that periodically request sensor readings. Figure 4.16 illustrates the configured traffic pattern of our cross-traffic with each burst consisting

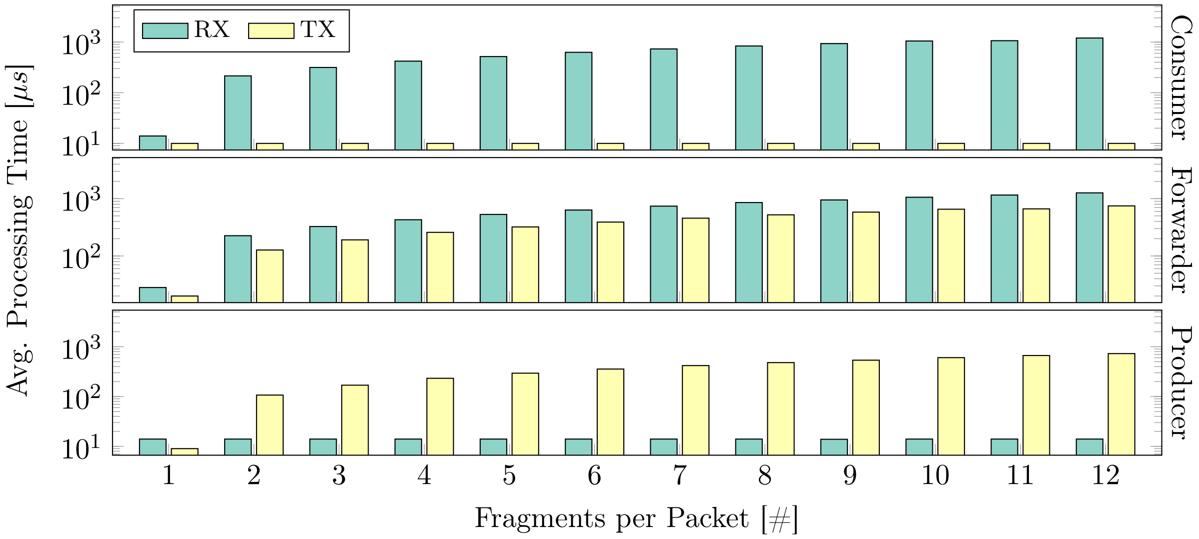


Figure 4.15: Average processing overhead for fragmentation and reassembly in a multi-hop scenario.

	CoAP	NDN	ICNL
Producer	93.53 %	94.05 %	94.40 %
Consumer	73.25 %	75.98 %	93.06 %

Table 4.2: Packet Reception Ratio (PRR).

of 200 UDP packets in succession with a 5–15 ms delay in between each transmission and a radio silence interval of 500–1500 ms. It is worth noting that our cross-traffic only adds wireless interference as the IEEE 802.15.4 frames are not delivered to the devices due to automatic MAC address filtering in hardware of the radio module. We further disable CSMA/CA and frame retransmissions of the radio transceiver to explore reliability gains without distortions of automatic corrective actions in hardware. Indeed, disabling such hardware features in real deployments is not a far-fetched scenario as sophisticated link layer protocols that employ time-slotted algorithms must satisfy strict time constraints, while CSMA/CA and retransmissions lead to indeterministic and drifting media accesses. In fact, several radio transceivers that dual operate IEEE 802.15.4 and Bluetooth Low Energy do not even support such mechanisms in hardware, but rather rely on software implementations.

Table 4.2 lists the packet reception ratio (PRR) for consumer and producer roles using the Name_{long} configuration. For each deployment the number of received requests on the producer lies within ≈ 93–94 %. Conversely, the percentage of successfully received responses on each consumer varies between ≈ 73–76 % for CoAP and NDN and ≈ 93 % for our ICNLoWPAN operation.

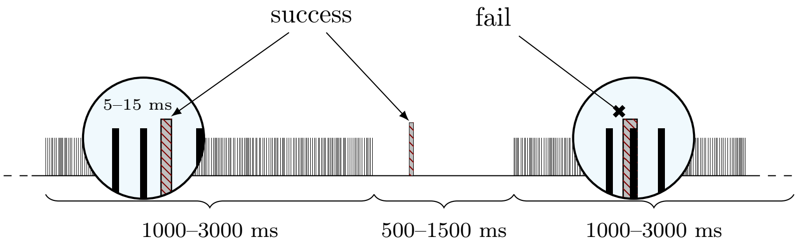


Figure 4.16: Radio interference due to bursty cross-traffic.

The performance gain of ICNLoWPAN results from strongly compressed packets which lead to a significantly reduced on-air time for the low power wireless transmission (see Figure 4.11). This reduces collision probability with interferer traffic, especially when responses are sent during a burst as shown in Figure 4.16. Furthermore, responses of a producer always follow the successful reception of a request. Reduced transaction times with ICNLoWPAN leave more time to the next interferer transmission within a burst, which further reduces the probability of overlapping cross-traffic compared to the NDN and CoAP operation.

4.4 Conclusions

IoT networking has proven to benefit from information-centric architectures in several directions. In this chapter, we worked out the components for adapting NDN to a LoWPAN edge: encoding, compression, framing, and fragmentation. By leveraging the NDN stateful forwarding for compression on path, we could again take particular advantage of the information-centric approach in low power lossy IoT networks.

Theoretical estimates and extensive measurements of our protocol implementation on IoT hardware revealed these benefits in comparison with plain NDN and the IP world (CoAP over 6LoWPAN). Our experimental results clearly showed that ICNLoWPAN outperforms NDN and CoAP in terms of media utilization as well as energy consumption. ICNLoWPAN further reduces end-to-end latencies in multi-hop scenarios, and contributes to an improved reliability in lossy environments while preserving battery resources. Depending on the use case, savings typically range from 20 % to 33 %. With these results, we hope to contribute insights to the community, to substantiate corresponding standard development [33], and to encourage deployment of NDN in the constrained IoT.

Chapter 5

Quality of Service for ICN

Abstract

The Internet of Things (IoT) comprises a relevant class of applications that require Quality of Service (QoS) assurances. Information Centric Networking (ICN) has shown promising characteristics in constrained wireless networks, but differentiated QoS has not yet fully emerged. In this chapter, we design and analyze a QoS scheme that manages the NDN resources *forwarding and queuing priorities*, as well as the *utilization of caches* and of *forwarding state space.* In constrained wireless networks, these resources are scarce with a potentially high impact due to lossy radio transmission. We explore the two basic service qualities (*i*) *prompt* and (*ii*) *reliable* traffic forwarding. We treat QoS resources not only in isolation, but correlate their use on local nodes and among network members. Network-wide coordination is based on simple QoS code points that can be distributed via a routing protocol. Fairness measures that prevent resource starvation are part of this management scheme. Our findings indicate that our coordinated QoS management in ICN does not only effectively prioritize the privileged data chunks, but also improves regular data communication. We can show that appropriate QoS coordination can enhance the overall network performance by more than the sum of its parts and that it exceeds the impact QoS can have in the IP world.

5.1 The Problem of Distributed Resource Management in ICN

Implementing service differentiation and assurance in a network raises the challenge of managing distributed resources without sacrificing them. Common Internet approaches follow a flow-based (*e.g.,* IntServ) or a class-based (*e.g.,* DiffServ) concept. A flow-based and admission-controlled resource reservation requires dedicated signaling and state, which quickly reaches scalability limits with resource exhaustion. In the presence of ubiquitous in-network caching and request aggregation, content endpoints are unspecified for ICN and data paths are inherently multi-source, multi-destination, and possibly widely disjoint. This makes ICN flows difficult to identify and to maintain.

Resource allocation according to packet classes requires ingress shaping and filtering, since

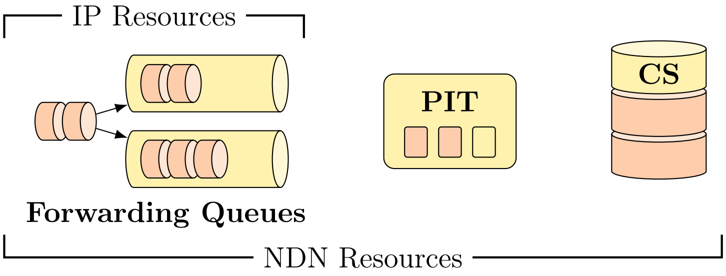


Figure 5.1: Manageable resources in IP vs. NDN.

unforeseen traffic bursts quickly exhaust the per-class reservations and counteract service assurances. In several ICN flavors including NDN and CCNx, link occupancy and forwarding demands are steered hop-by-hop in a request-response fashion. Small requests trigger data replies of unknown size, provenance, and timing. This complicates reliable resource predictions for responses in NDN. Shaping and dropping Interests can prevent resource exhaustion, but may leave the network underutilized. Restricting ingress only to data may lead to bursts of unsatisfied Interests, which waste network resources.

In-network caches in ICN enrich the field of manageable resources. Caches reduce latency and forwarding load and often take the role of a (large, delay-tolerant) retransmission buffer. With NDN/CCNx, additional resources come into play in the form of Pending Interest Tables (PITs) that govern stateful forwarding. The overall resource ensemble is visualized in Figure 5.1 and raises various issues. Capacities in forwarding, caching, and pending Interest state may be largely heterogeneous. A wirespeed forwarder may supply negligible cache memory compared to its transmission capacity, for example. In the IoT the opposite is often true in that flash memory is normally shipped in 'infinite' sizes when compared to the main memory (PIT) or the wireless data rate. A beneficial resource management faces the problem of how to carefully balance these resources and arrive at an overall optimized network performance [126].

Resource complexity, however, extends beyond a single system. The impact of distributed resources is easily flawed if management cannot jointly coordinate contributions. Neighboring caches, for instance, are less effective if filled with identical copies. The effective overall cache capacity of the system can be increased by using cooperative caching strategies, which are needed in particular if caches are too small to hold the requested amounts of data during their validity periods. A more delicate problem arises from PIT state management. If neighboring PITs diverge and no longer represent common forwarding paths (see Figure 5.2) all data flows terminate and forwarding resources are wasted. This problem of state decorrelation was first reported in [127].

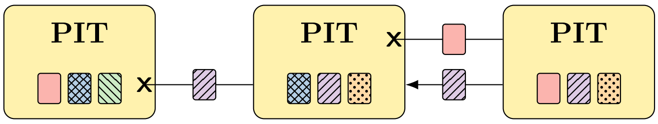


Figure 5.2: PIT decorrelation terminates data paths.

5.1.1 QoS Qualifiers

QoS extensions for ICN have recently attracted attention and generated various efforts within the IRTF ICN research group [128, 129, 130, 131, 132]. A proposed flow classification mechanism [130] differentiates traffic flows based on content name prefixes using two different methods. The first method, *EC3*, introduces a new message header entry in Data packets that indicates the prefix length of the content name characteristic for the classification process. Once a consumer learns about such an equivalence class, it can also include equivalence class indicators into subsequent Interests. The second method, *ECNCT*, encodes classification indicators directly into names at content creation using a new type of name component. This solution does not inflate messages with additional headers, which is advantageous for constrained IoT deployments. Nevertheless, encoding flow classification indicators in typed name components leads to an inflation of names in the routing system. Identical content published with different classification values will lead to duplicate names that cannot be aggregated in Forwarding Information Bases (FIBs). Analogously, the default matching functions used for the Pending Interest Table (PIT) and Content Store (CS) will consider names that differ only in their flow classification as dissimilar, which conflicts with Interest aggregation and cache utility.

In contrast, our approach to flow classification is simpler and leaner. We attribute QoS service classes to name prefixes and distribute this information to the ICN node (*e.g.,* via a routing protocol) to be kept as persistent state. This approach omits packet overhead, remains compliant with Interest aggregation and caching, but cannot process content of the same name in different service classes.

In the Internet Research Task Force (IRTF), two approaches to QoS treatment are under discussion. An end-to-end QoS framework [131], in which non-routable QoS markers are appended as suffixes to content names. In analogy to DiffServ, these markers are then used to apply different resource allocation mechanisms to the request and response messages. The FIB is extended to ignore QoS markers, and the PIT is modified to disaggregate pending requests that have differing QoS markers. This disaggregation may lead to PIT inflation in particular for setups with a high diversity of QoS markers. This work has also been presented to the IRTF [132].

Tsilopoulos *et al.* [133] identify three different types of traffic based on two characteristics in ICN traffic. The authors introduce two extensions to CCN in order to handle these traffic types, *Persistent Interests* and *Reliable Notifications*. Persistent Interests are valid for Data packets which are produced during a pre-defined period of time. Reliable Notifications inform receivers that real-time data is available and are propagated reliably on a hop-by-hop basis. Notifications that are not acknowledged in time are retransmitted. If a receiver successfully receives a notification but does not receive data in a given time, new Interests are created to renew the request.

5.1.2 Distributed Forwarding Resources

MIRCC [134] introduces a rate-based, multipath-aware congestion control scheme for ICN. Each Data message in MIRCC contains a rate value which in turn is used to calculate per-link rates. It is inspired by the Rate Control Protocol (RCP) [135] for IP networks.

Al-Naday *et al.* [136] manages forwarding resources and is experimentally evaluated for the PURSUIT architecture. This work implements a QoS differentiation scheme, where each forwarder manages virtual links that include packet queues with varying traffic rates and a designated traffic shaper. QoS information is encoded into the names and determines the mapping of traffic flows to low or high priority virtual links.

5.1.3 Distributed Cache Management

Caching policies that employ heuristics to inform a caching decision instead of caching all incoming content can be broadly organized into a number of different families, depending on what information they use to reach their caching decision.

The easiest way to achieve higher cache diversity without increasing the complexity of the caching policy is to cache probabilistically. The static version of this approach, commonly known as $Prob(p)$, uses a static probability p that governs how likely a given node will cache a given content chunk. It has been shown [137, 138] that $Prob(p)$ outperforms the default strategy of caching everything in terms of cache diversity, and that lower values for p correlate with higher diversity [137, 138, 139, 140, 141, 142].

Instead of using the same static caching probability for all incoming content, a caching strategy may also dynamically compute a probability for each individual node or even for each content chunk in order to adapt the caching behavior to the state of the network. These strategies may be based purely on node-local information, such as CS saturation or battery levels; on properties of the incoming content, such as its name, freshness, type, or producer; or on information from the wider network, such as the position of the caching node in the network topology or the CS contents of neighboring nodes. Examples include *ProbCache* [139], which computes the caching probability of a given content chunk based on the total number of hops between its producer and the consumer that requested it, and $pCASTING$ [138], which considers the freshness of the content as well as the node's battery level and CS saturation when calculating p. Both strategies have been found to increase the cache hit ratio, reduce the average number of hops required to hit requested content, and reduce the number of cache evictions [143]. Various other dynamic probabilistic caching strategies have been proposed, with decisions based on content freshness [138, 144], content popularity [145], or whether the content is already in a neighboring CS [146].

Not all caching strategies use the probabilistic approach. Instead, some exploit knowledge about the network topology [147, 148, 149, 150, 151, 152], which has the advantage of taking global knowledge about the network into account but often comes at significant costs such as lengthy setup times, communications and memory overhead, and vulnerability to changes in the topology [152]. Another class of caching policies has nodes cooperate with their neighbors, either explicitly by exchanging information [153] or implicitly by using pre-defined rules [154, 155, 156].

5.2 QoS Building Blocks for NDN

5.2.1 Distributed Traffic Flow Classification

General purpose networks simultaneously host competing traffic flows that exhibit varying resource requirements and time constraints. This also holds for typical IoT deployments, in which flows originate from sensors and actuators, or from remote cloud services that connect via gateways to the IoT domain. A flow classification is necessary for differentiating packets that belong to separate message flows, whenever the network should allocate distinct resources and treat them in a differentiated manner.

In the IP world, traffic flows are defined by the application endpoints and identified by address/port tuples (IPv4) or addresses plus flow label (IPv6). Since ICN abandons the host-centric paradigm, this definition of traffic flows no longer holds. In the presence of delocalized content and in-network caching, the concept of application endpoints becomes meaningless.

The characterizing property of these packets are content names (or prefixes). In the example of CCNx and NDN, content names appear in related Interest and Data packets. This ubiquity of names and their potential to impose hierarchies on content make them a distinguished component for identifying flows. Accordingly, we propose a traffic flow classification mechanism for NDN that builds on hierarchical names, prefixes, and longest common prefix match. It is explicitly designed to not put a strain on typically resource constrained IoT devices. In this regard, our scheme is computationally simple and does not require an additional overhead in message headers.

In this work, we consider service differentiation with respect to two quality dimensions: *latency* and *reliability*. For simplicity we only use a plain distinction in each quality, which results in a matrix breakdown of service levels as shown in Figure 5.3. More sophisticated differentiations apply analogously.

Service classes are assigned to flows according to a list of prefixes that are marked with a traffic class and maintained by each node, *e.g.,* as part of the Forwarding Information Base (FIB). Incoming Interest and Data messages are then mapped onto traffic classes by applying

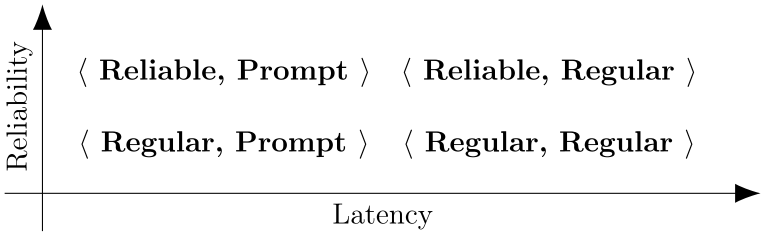


Figure 5.3: QoS Service Levels.

Traffic Class	Priority
/MO/ACM/ICN	<Reliable, Regular>
/MO/ACM/ICN/site/A/alarm	<Reliable, Prompt>
/MO/ACM/ICN/site/B/temp	<Regular, Prompt>

Table 5.1: Traffic classes and appropriate priority mappings.

a longest prefix match against this list of traffic identifiers. Traffic class names may look as in Table 5.1.

Given this example, Interest and Data messages containing the prefix */MO/ACM/ICN/site/C* would map to the closest valid traffic class. In this case, the longest matching prefix is */MO/ACM/ICN*.

This work requires such prefix marking to be deployed at all nodes within a network domain. The distribution and maintenance of QoS configurations may be performed by a regular routing protocol or by more specific QoS policy management protocols. To illustrate the first, we recall that any NDN routing protocol distributes name prefixes across the nodes of its domain to enable a dynamic name-based routing. Prefix (or name) advertisements could be easily augmented by QoS qualifiers so that QoS configurations would allow for actualization within one update cycle of the routing protocol in use. Nevertheless, a general discussion of QoS signaling is beyond the scope of this work, which focuses on a lightweight flow-to-priority mapping for the purpose of quantifying the benefits of introducing QoS-sensitive resource management to ICN-based networks.

It may be noted that fine-grained service differentiation within a complex name hierarchy can result in large QoS prefix tables that may inflate the FIB. For the constrained IoT use case, though, we argue that sensor readings and actuator settings are machine type communications (MTC) with (short) names configurable according to processing needs. Hence QoS overheads should easily comply to resource constraints.

5.2.2 Manageable Resources

Forwarding onto the Link Layer

The link layer manages access to the media and provides space to buffer packets. In low-power wireless networks, media access times are highly susceptible to media saturation and buffer spaces are small. While time slotted technologies such as the IEEE 802.15.4e TSCH mode and Bluetooth Low Energy access media in a deterministically scheduled manner, the unslotted CSMA/CA version of IEEE 802.15.4 or long-range radios such as LoRa are more susceptible to packet collisions among neighboring nodes.

In the IoT, content producers that generate sensor readings may produce egress traffic at a rate that is much higher than the average media access time. In addition, nodes may further need to forward ingress traffic in multi-hop scenarios. For this purpose, buffering egress traffic is necessary to cope with traffic spikes.

Queuing and buffering take the same role in ICN as in the IP world. Class-based forwarding queues will process packets of the *prompt* flow class before packets with a regular priority while buffer space will prevent packet drops. It should be noted, though, that rapidly forwarded packets in NDN will also quickly satisfy PIT entries and thereby free forwarding resources for unprioritized traffic on the same path.

Pending Interest Table

The Pending Interest Table (PIT) enables the stateful forwarding plane of CCNx and NDN and thereby governs the flows in the network. The size of the PIT resource effectively dictates the maximum number of simultaneous open requests and the coherence of PIT entries along a path determines whether flows can propagate without barriers. In normal NDN operation, PIT state is allocated when an Interest message is processed, and it is removed in two scenarios. Either a returning Data message consumes the PIT state, or a timeout after a succession of retransmissions clears the state.

PIT saturation is likely to occur in IoT deployments. Limited RAM resources, slow processing power, delayed media access, and packet loss in low-power networks with intermittent connectivity can all cause the PIT to reach its maximum capacity.

The typical way of handling incoming Interest messages at a saturated PIT is to drop them in order to avoid cancelling active but incomplete request operations. The penalty for dropping such Interests is an increase in latency due to retransmissions, which usually happen on the scale of seconds. To avoid high latencies for time-sensitive traffic flows, a PIT eviction strategy is necessary, which accounts for (*i*) unhindered forwarding of prioritized Data, and harmonizes with (*ii*) the retransmission mechanisms of the regular NDN.

Content Store

Due to memory constraints, the Content Store (CS) is typically small, necessitating heuristics for deciding what content to cache at which node instead of indiscriminately caching all incoming content. There is a wealth of existing research on how to make the most efficient use of limited CS space, with a number of different strategies employing various heuristics to decide whether or not to cache incoming content (see Section 5.1.3). This aspect of caching is called the *caching decision strategy*.

The introduction of traffic flow priorities adds an additional dimension to the caching decision. Regardless of which specific caching decision strategy is employed, content marked as *reliable* should always be cached as it is imperative that this content is available throughout the network. Thus, reception of *reliable* content should not trigger the caching decision strategy; instead, control should be handed directly to the cache replacement strategy (see below). The question whether *prompt* content should be cached with a higher priority than content with *regular* latency requirements is not as clear-cut. Caching *prompt* content with higher priority would have a positive effect on future transmissions of that content object (either by retransmission of the original request or by new requests) and thus have a positive effect on latency, although the potential gain in this aspect is dependent on path length. Any content that is marked as *regular* in both QoS dimensions should be treated as normal; in other words, the caching decision strategy is consulted.

After a node has decided to cache a new content object, an additional step may have to be taken in case the CS is at capacity. This aspect of caching is the *cache replacement strategy*. In most cases, CS contents will be replaced using a simple heuristic such as Least Recently Used (LRU). However, once again the introduction of traffic flow priorities adds an additional dimension to this decision.

In general, incoming content should not replace content of a higher priority. Therefore, content with *regular* latency requirements should not replace *prompt* content and content with *regular* reliability should not replace *reliable* content. When it comes to the correlation between latency and reliability, the primary goal of the CS should be to ensure content availability, which places a stronger emphasis on the reliability aspect. Thus, *reliable* content with *regular* latency should be able to replace *prompt* content if no other content is eligible to be replaced. If all content is of the same priority class, regular replacement rules (e.g., LRU) should apply.

In probabilistic caching, as introduced in Section 5.1.3, each node caches incoming content according to a certain probability p. Regardless of how exactly p is determined (whether statically or by one of the dynamic methods discussed in Section 5.1.3), the probabilistic approach may be refined by differentiating between two separate probabilities p_{reg} for regular content and p_{rel} for reliable content, with $p_{rel} > p_{reg}$. This has the effect that a CS at each node will have different contents, thus contributing to CS diversity across the network by making a larger

Isolated Decisions	Resource Correlations	Joint Prioritization
Forwarding Queue	**PIT–CS Correlation**	**PIT Coherence**
Expedite *prompt* traffic	If *Prompt* Data has no PI → cached with priority	Aligned node decisions *regular* < *reliable* < *prompt*
Pending Int. Table	**Fwd–CS Correlation**	**CS Efficiency**
Evict *regular* for *prompt*	If *Prompt* Data drops → cached with priority	Aligned node decisions *regular* < *prompt* < *reliable*
Content Store		
Evict *regular* for *reliable*		

Table 5.2: Classification of QoS mechanisms and decisions.

range of content available as cached copies, while giving consideration to service classes ensures that higher-priority content is still treated preferentially.

5.2.3 Distributed QoS Management

We are now ready to present our approaches to distributed resource management for supporting QoS in ICN. The corresponding mechanisms fall into three classes, depending on the level of interdependence between resources on the same device or between devices. A summarizing table is further given in Table 5.2.

Locally Isolated Decisions

The straightforward allocation of independent resources to packet forwarding follows three simple rules:

Prioritized forwarding applies to flows marked as *prompt*.

Cache (re-)placement decisions obey the priority order *reliable* (highest) to *regular* (lowest). In the presence of probabilistic caching strategies, the weights are set accordingly.

PIT management operates in favor of rapid packet forwarding, so PIs enter the PIT in the order *prompt* (highest) to *regular* (lowest). Newly arriving Interests that meet a PIT saturated with entries of equal or higher priority will be dropped. Otherwise, a premature PIT eviction would lead to incoherent PIT states along common forwarding paths (see Sec. 5.1).

Local Resource Correlations

These are decisions that entail interaction between mechanisms on the same device (intra-device correlations). This includes the correlation between the caching decision and cache replacement strategies, where, *e.g.,* the caching decision may pre-empt the cache replacement decision in the case of *reliable* flows, while the cache replacement decision may drop content even in the case of a positive caching decision if no content of the same or lower service class can be replaced. In detail we take the following steps:

with PIT entry: If arriving Data meets a valid PIT entry, Data is forwarded according to priorities *and cached* with priority, if marked as *reliable*. In the case of exhausted prioritized forwarding queue, *prompt* traffic will be cached with the highest priority.

without PIT entry: If arriving Data meets no valid PIT entry, cache placement will still be initiated for *prompt* and *reliable* data in subsequent order. For probabilistic caching, weights are adjusted accordingly.

In balanced, unconstrained NDN networks, returning regular Data meets open PIT states. For saturated PITs, however, PIT entries may time out quickly, or resource management may enforce eviction of PIT entries in favor of other requests. Allowing Data without corresponding PIT entries to be cached may introduce the threat of cache poisoning attacks. However, a simple rate limiting on incoming Data packets and a reduced cache time for these CS entries may reduce the attack surface in our constrained environment. Further analysis of related effects is left to future work.

Joint Resource Prioritization

Such mechanisms affect resources across multiple or all devices in the network (inter-device correlations). These include maintaining PIT coherence by ensuring that all nodes apply uniform QoS mechanisms when replacing content of different service classes, as well as achieving CS diversity by introducing probabilistic caching based on priority classes. In our system, a joint resource prioritization is achieved as follows.

PIT coherence is increased by applying the same PIT eviction strategy at all nodes, *e.g.,* evict *regular* before *reliable* before *prompt*.

Cache efficiency increases with probabilistic caching using aligned configurations of equal cache weights at all nodes. It is noteworthy that probabilistic caching reduces the risk of starvation for low priority content due to higher cache diversity, even if high priority flows dominate the network.

A summary of the different QoS decisions while processing Interest and Data messages are visualized in a flow description in Figure 5.4.

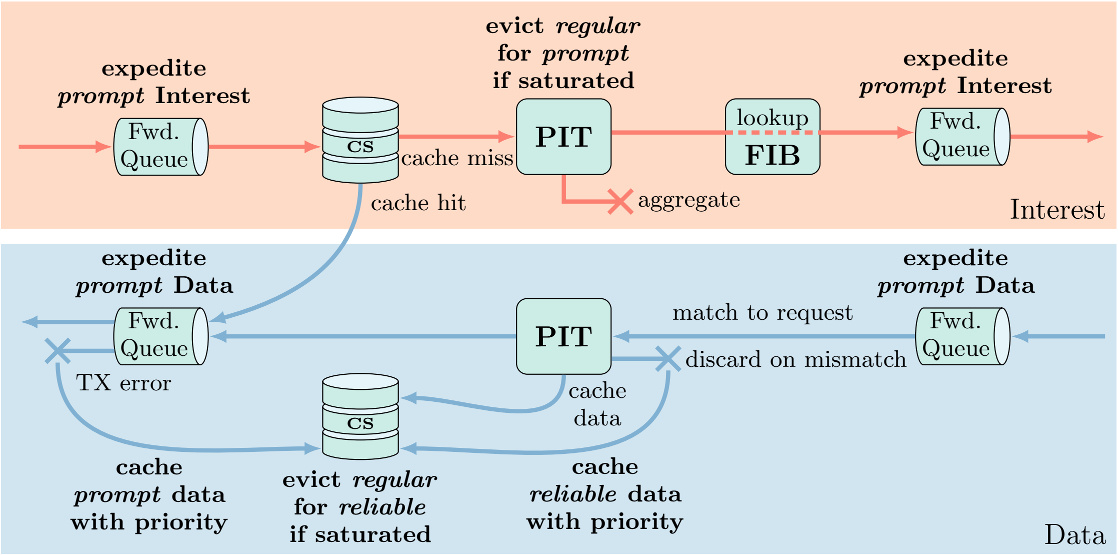


Figure 5.4: Flow description for Interest and Data messages in a QoS enabled NDN forwarder.

5.3 Evaluating Competing Traffic

5.3.1 Implementation and Experiment Setup

Testbed. We conduct our experiments on the FIT IoT-Lab testbed using typical class 2 [1] IoT devices that feature an ARM Cortex-M3 MCU with 64 kB RAM and 512 kB ROM. Each device further contains an Atmel AT86RF231 2.4 GHz transceiver [69] to operate on the IEEE 802.15.4 radio.

QoS aware nodes. All devices run on RIOT OS [157] version `2019.04` with the integrated NDN network stack CCN-lite [106], which we extended with our QoS management scheme. In addition to the PIT and CS management strategies, a very lightweight prioritized forwarding was implemented using a single packet double-buffer that allows for pairwise packet re-ordering. We also note that network stack performance usually exceeds link speed. While IEEE 802.15.4 provides a theoretical maximum of 250 kbit/s, I/O and data processing in the network stack is at least one order of magnitude faster [29].

System parameters. Following the large-scale deployment in [111], we configure a maximum number of four retransmissions and a retransmission interval of two seconds for CCN-lite. To analyze our approach under different levels of network saturation, we configure the maximum capacities of PIT and CS to range between 5 and 30 elements. Notably, the estimated RAM usage for 30 PIT entries and 30 CS elements with name lengths of $\approx$ 32 bytes is already approximately 11 KiB, which is around 17% of the total available RAM for our hardware platform.

Caching parameters. We consider the caching procedure to consist of two fundamental

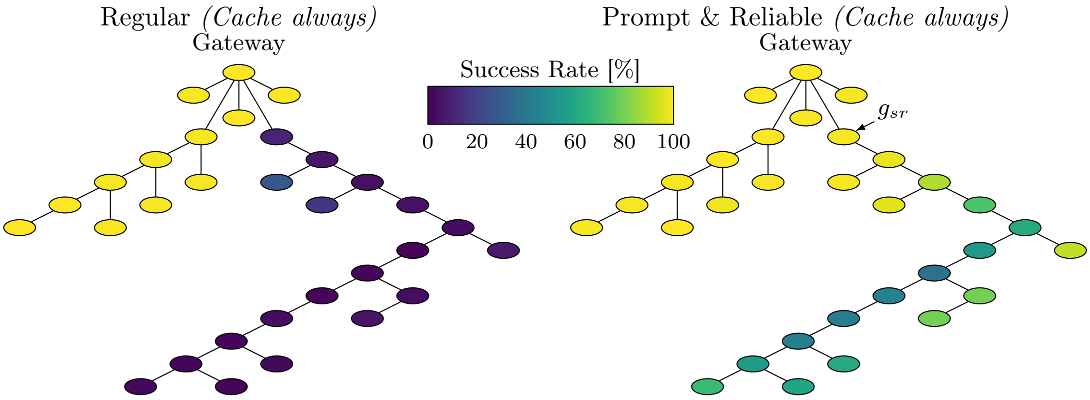


Figure 5.5: Nodal success rates for *Scenario 1* using regular traffic (left) and *reliable* actuator traffic (right).

steps: (i) caching decision, and (ii) cache replacement. In our experiments, we use two different caching decision strategies and one cache replacement strategy.

The first decision strategy is to *cache always* incoming Data packets, with the restriction for regular traffic that Data packets without a corresponding PIT entry are dropped and not cached. For *reliable* traffic, we adjust this strategy, such that Data packets without PIT entries are still considered for caching. The second decision strategy is to *cache probabilistically*. Every node caches incoming Data packets with a probability p_{reg} of 30% for regular traffic, and with a probability p_{rel} of 70% for *reliable* traffic. For a saturated CS, our cache replacement strategy evicts content store elements using the *least recently used (LRU)* policy.

5.3.2 Topology Setup

The testbed provides access to multiple sites with varying numbers of M3 IoT devices and network characteristics. We deploy our applications on the *Grenoble* site, as this supports significantly complex multi-hop paths. We choose 31 M3 devices to host a rich network topology with varying fan-outs, chains, and branch sizes. As such, this single topology represents well a non-trivial IoT edge network and therefore facilitates the reproducibility of our results. In this topology, one device acts as a gateway node and the other 30 devices act as sensors and actuators. Since convergecast is the most predominant traffic pattern in common IoT scenarios, we arrange our devices to form a *Destination Oriented Directed Acyclic Graph (DODAG)* that is rooted at the gateway node. Approximately 60% of the nodes are reachable from the root within 4–5 hops, while the remaining devices have path lengths up to 12 hops. The topology is visualized in Figure 5.5. Extended left and right wings in the routing topology result from long hallways in the *Grenoble* site.

5.3.3 Experiment Scenario 1: Mixed Sensors and Actuators

We want to quantify the efficiency of QoS enhanced forwarding in challenged multi-hop deployments that display typical traffic patterns. With this in view, we analyze our approach first in a scenario with mixed traffic of unprioritized sensor readings and prioritized actor commands.

The gateway node requests temperature readings from the 30 sensor nodes every 10 s with ± 2 s jitter interval. Thus, on average, the request rate at the gateway approximates to 3 *packets/s* and including the reception rate of responses, the gateway handles 6 *packets/s*. The naming scheme for each request consists of a prefix, a device-specific *node id*, and an increasing sequence number. We refer to this traffic equally as *sensor readings* or *sensor traffic*.

In addition, all 30 devices further act as actuators that periodically request a device-specific state from the gateway node every 5 s with ± 1 s jitter. This yields a request reception rate of 6 *packets/s* on the root node. The naming scheme for these requests similarly consists of a prefix, a device-specific *node id*, and an increasing sequence number. We name this traffic *actuator traffic*.

In this scenario, sensor and actuator data are device-specific and thus only scattered destinations benefit from on-path caching during the narrow window of Interest retransmissions.

5.3.4 Results

We measure and record the *success rates*, *goodputs* and *time to completion* for each node in the topology, while we analyze the network utilization for gateway and actuator traffic separately with and without the proposed QoS features enabled.

Success rates. In our first experiment we focus on the nodal success rates using the first scenario with a PIT and CS limitation of 5 entries. The gateway is configured with a PIT limitation of 50. Figure 5.5 shows the resulting success rates for (*i*) the regular operation of NDN on the left-hand side, and (*ii*) a setup with prioritized actuator traffic using the *prompt* and *reliable* QoS service levels on the right-hand side. The success rates per node are color coded and range from 0% (purple) to 100% (yellow).

Figure 5.5 clearly depicts huge differences in success rates for both configurations. In the normal NDN operation, nodes close to the gateway, as well as the left wing of the topology, achieve 100% success rates. Strikingly, the right wing exhibits major network stress, with nearly all actuators having a success rate below 10% due to PIT overflows. Conversely, with QoS service levels enabled, the right wing shows much enhanced network performance, which results in overall higher success. With this configuration, 70–100% of the packets arrive at actuators close to the gateway, and 40–70% at more distant nodes. Leaf nodes farther away show a greater improvement in success rates than forwarding ancestor nodes.

This striking example nicely illustrates the positive effect of PIT coordination obtain from QoS differentiation. While in the regular case Interests at nodes with exhausted PIT are discarded as they randomly arrive, the QoS marking preselects those requests that are prioritized throughout

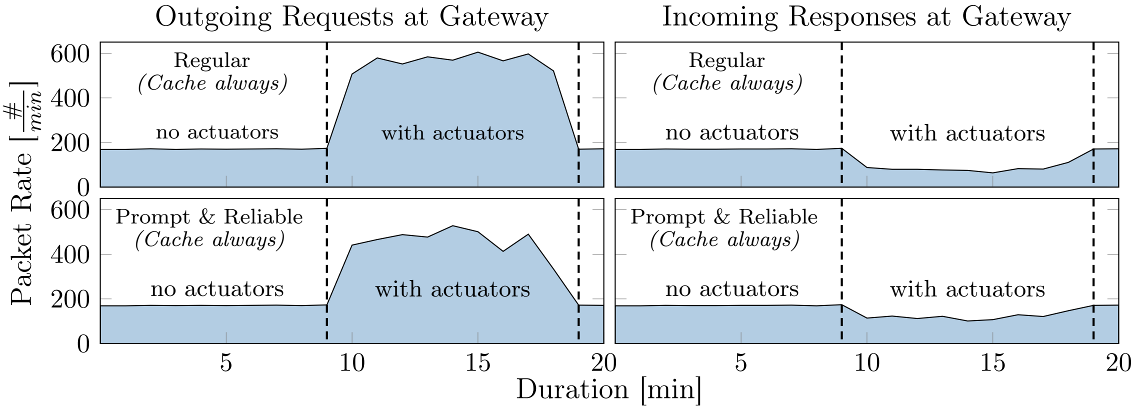


Figure 5.6: Packet transmission rate per minute for requests and responses with and without (prioritized) cross traffic measured at the gateway.

the network, leading to fewer retransmissions and a more efficient use of the overall PIT space available in the network. A more detailed analysis that compares with enhanced caching effects will be discussed in Section 5.4 and is visualized in Figure 5.10.

Network utilization. In typical convergecast settings, a majority of the traffic traverses the gateway to reach remote endpoints. Thus, a robust operation of the gateway node is crucial to ensure adequate performance of the entire IoT network. Due to the increasing sequence numbers used in the naming scheme for *Scenario 1*, virtually no response contributes to future in-network cache hits. To gauge the network load on the gateway, we analyze the number of outgoing requests and incoming responses during a setup in which actuator traffic is added to the traffic from the gateway after approximately eight minutes. We first perform this experiment without any QoS features enabled, and then repeat it with the adjustment that actuator traffic is prioritized using the *reliable* and *prompt* service levels. The PIT of the gateway is configured to a maximum of 50 entries, while the remaining nodes have a PIT maximum of 5 entries. The CS is limited to 5 entries for all nodes.

We observe in Figure 5.6 that the sensor traffic exhibits a steady request-response flow of about 180 *packets/min* for both requests and responses. As soon as the actuators initiate their periodic requests (at minute eight), the network load at the gateway increases. The number of requests spikes threefold due to network layer retransmissions, while the number of returning responses drops to half. In contrast, the setup with prioritized actuator traffic clearly admits a reduced number of requests at the gateway while achieving a higher response rate.

These results reveal that prioritizing the actuator traffic has a positive effect on the overall network load due to reduced retransmissions.

Goodput. Figure 5.7 shows that while PIT sizes have a significant impact on the goodput at the gateway and at each actuator during normal operation, this effect is reduced when QoS service classes are introduced. When using service classes and the *always* caching decision strat-

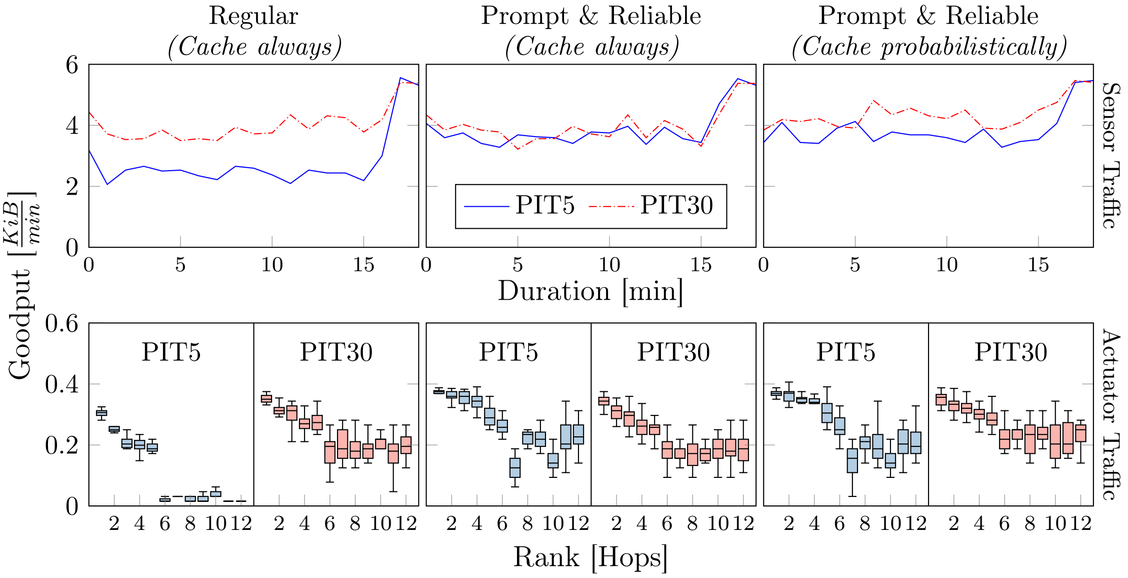


Figure 5.7: Goodput for *Scenario 1* with actuator and gateway traffic (CS size of 5).

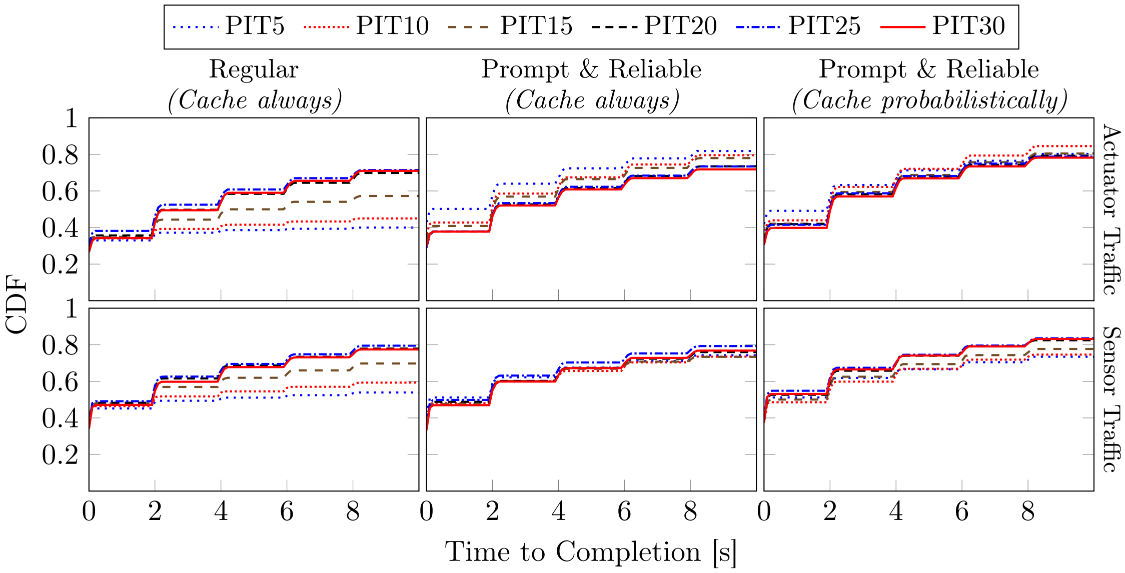


Figure 5.8: Completion time for *Scenario 1* with actuator and gateway traffic (CS size of 5).

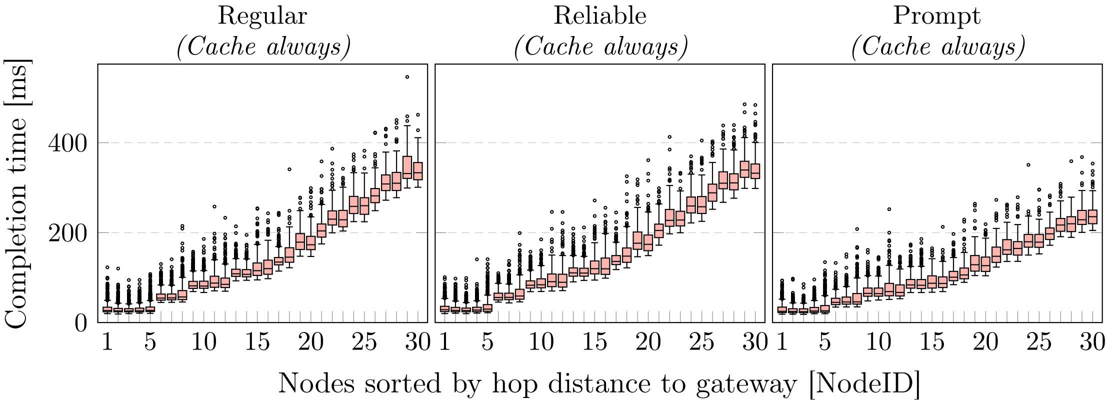


Figure 5.9: Time to completion per actuator and quality dimension in *Scenario 1* using PIT and CS sizes of 5.

egy, network members with small PIT sizes can reach a level of goodput that is comparable to those of the largest PIT size during normal operation. With *probabilistic* caching in operation, the network performance increases even further for larger PITs, which support a higher number of concurrent flows that can leverage cache diversity better. This overall enhanced transport performance is caused by flows that complete with delay based on segment-wise available network capacities and retransmissions. The corresponding temporal effects can be observed from Figure 5.8.

At actuator nodes far from the gateway, our QoS mechanisms mitigate the effects of PIT exhaustion, which in normal operation leads to an abrupt collapse of the throughput at around a rank of 6. Our QoS mechanisms cause a smooth, gradual decline in performance instead.

Time to completion. We can see from Figure 5.8 that content arrival times are significantly reduced for smaller PIT sizes when QoS service classes are introduced—for traffic at gateways and even more at the prioritized actuators. Simultaneously, we see again a signature of enhanced traffic delivery for QoS-coordinated flows that shows doubled success rates from 40% to 80%. It is worthwhile to (re-)observe that the sensor traffic also experiences improvements due to a more efficient balancing of resources—small PIT sizes in particular.

We now quantify the content arrival times specifically for each quality dimension. Figure 5.9 illustrates the nodal content arrival times with PIT and CS sizes both set to 5. In all three displayed configurations, completion time increases with the distance to the gateway. Content arrival times range from 5 ms to 350 ms for the majority of *regular* requests, and the *reliable* configuration does not change this. In contrast, the *prompt* configuration yields noticeably faster delivery times for all nodes, in particular nodes far away from the gateway experience a reduction of about 100 ms (≈ 30%). This shows that the very light-weight priority queuing based on a simple double-buffer already shows an important impact throughout the constrained network.

5.4 The Impact of Caching

5.4.1 Experiment Scenario 2: Sensing and Lighting Control

In the second scenario, we change the role of actuators but leave the sensor readings unchanged. Instead of independent actuators that receive disjoint instructions, we envision a scenario of lighting in which groups of fixtures switch lights in a coordinated way. Hence, these groups receive identical commands and caching becomes applicable.

To explore the event space by mixing group memberships, our experiments proceed as follows. For each request, an actuator randomly joins one out of five possible 'lighting' groups. The naming scheme for such requests is changed to include the selected *group id*, instead of a device-specific *node id*. Besides naming, we use the same request parameters for the actuator traffic as in the first scenario. We repeat this process 240 times for each configuration in order to explore the state space of unevenly distributed groups and converge statistics.

In this scenario, prioritized actuator traffic flows from the gateway to multiple destinations and thus benefits from on-path caching. Accordingly, we expect the network performance to improve over that for Scenario 1.

5.4.2 Results

Success rates. The overall success results presented in Figure 5.10 confirm these expectations. With this figure we dig deeper into network reliability and examine the success rates for a range of PIT sizes (scenario 1) and Content Store sizes (Scenario 2) plotted against the node ranks.

Success in content delivery for the second scenario nicely approaches 100% in most QoS settings, while in contrast Scenario 1 experiences significant loss rates above 50% at the edges for all sizes of pending Interest tables. With caching even for the most constrained number of 5 PIT entries, an increase from 5 to 30 of the CS sizes suffices to turn traffic from QoS class *reliable* into a fully reliable service. While we do see some failures at higher ranks for the small CS size of 5 similar to *Scenario 1*, increasing the CS capacity to 10 is already sufficient to attain success rates above 80%. The most striking contrast is found in comparison to the results of regular NDN operation, where even with a maximum CS size of 30 the failure rate stays at 30–40% at higher ranks. Regular NDN traffic apparently profits less from cacheable content. This is mainly due to PIT decorrelation, which breaks content flows.

As previously observed, the success rate for *regular* actuator traffic collapses at higher ranks. The sensor traffic performs similarly poorly. Increasing the maximum PIT size gradually improves the nodal success rates. In addition to the *regular* traffic, we further show actuator traffic with *prompt & reliable* QoS service levels. Overall, the setups with prioritized traffic show great improvements for the smaller PIT sizes, while the *probabilistic* caching decision strategy performs slightly better than the *always* strategy. A surprising effect is observed with a PIT size

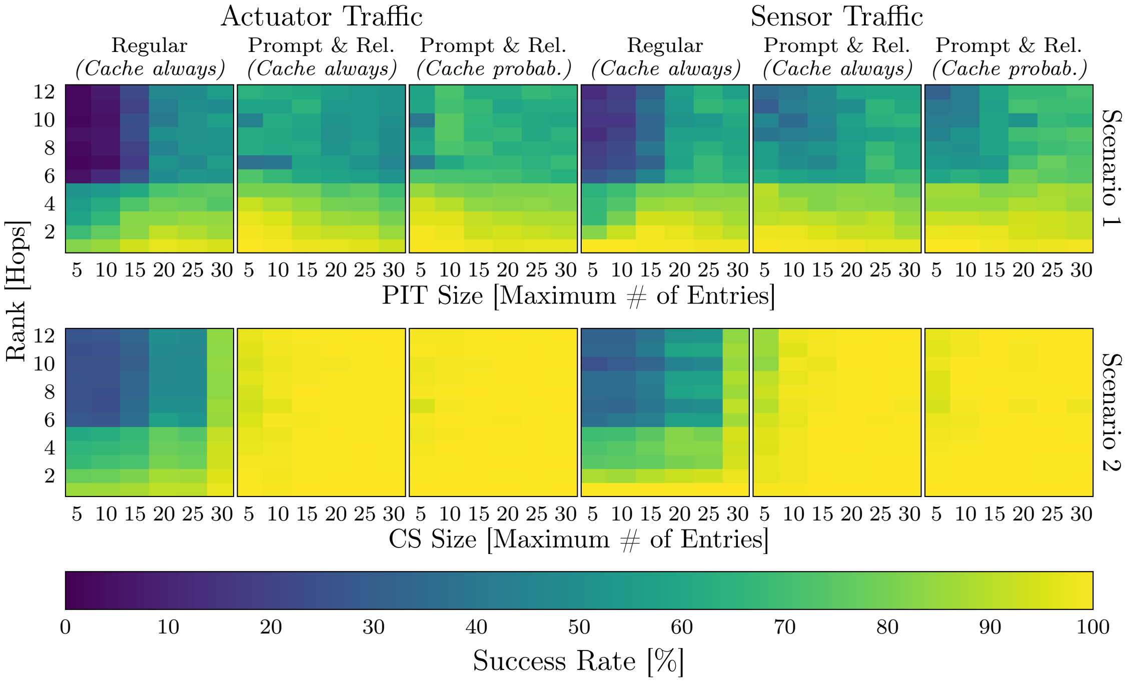


Figure 5.10: Success rates per rank for *Scenario 1* and *Scenario 2* using varying PIT and CS sizes.

of 30: the success rates for all configurations decline slightly for lower ranks. This is caused by an increased retransmission overhead per node, resulting in link saturation.

Time to completion. QoS service classes have a significant impact on the content arrival times for both the actuators and the sensors, as was already observed for *Scenario 1*. Figure 5.11 reflects the same qualitative picture for *Scenario 2*, but at significantly reduced probabilities of retransmission. The latter is due to actuator traffic that is coordinated in QoS classes and coherently serviced from caches—a large reduction in overall network load. Results slightly improve for enhanced cache diversity in probabilistic caching, with 90% of the packets arriving promptly ($<$ 100 ms) at cache sizes of at least 10 packets. Similar to *Scenario 1*, CS sizes become less relevant in the presence of QoS marking, since prioritized traffic arrives quickly at its destination and remains unaffected by regular cache replacement.

While we observe that increased CS sizes contribute to reduced completion times and improved success rates, we also notice that even a CS size of 30 does not suffice for the *regular* configuration. On the other hand, the configurations that use QoS service levels display close to 100% success rate, even with severely limited CS sizes, and about 70% of all requests for each traffic type complete in less than 100 ms.

Cache hits. Analyzing the cache efficiencies supports these observations. Figure 5.12 displays the relative in-network cache hits for actuator traffic in setups with varying CS sizes. While the regular NDN operation yields a marginally improving cache hit ratio for increasing CS

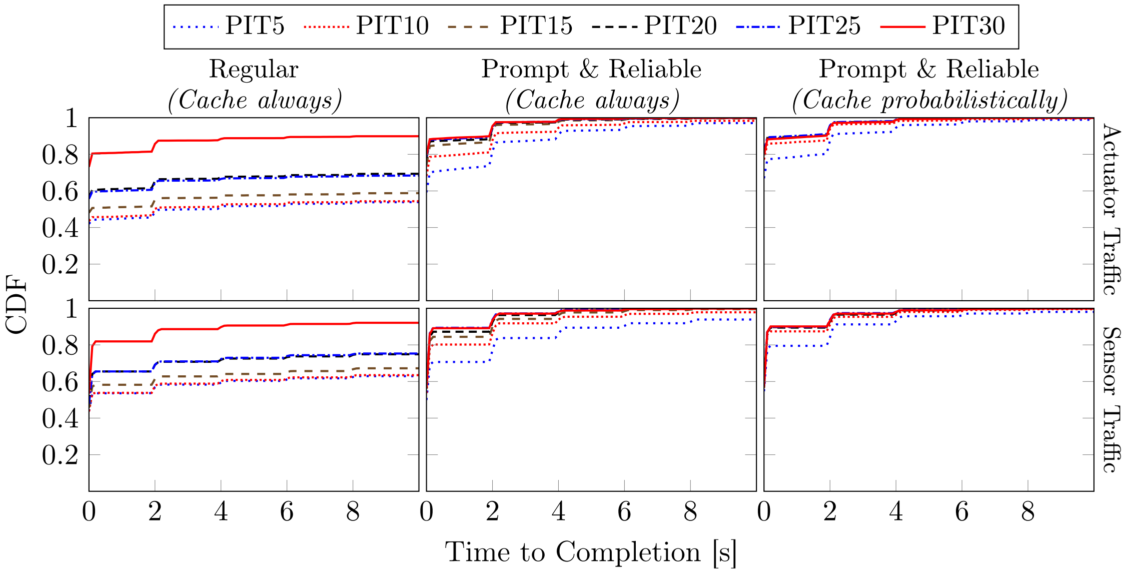


Figure 5.11: Time to completion for *Scenario 2* with actuator and gateway traffic using a PIT size of 5.

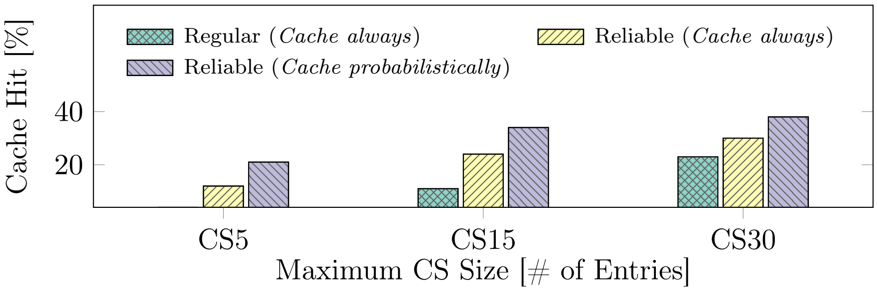


Figure 5.12: Cache hits for actuator traffic in *Scenario 2* with a PIT size of 5.

sizes, both QoS enabled setups exhibit a noticeable enhancement. This improved cache efficacy is caused by the privileged cache resource utilization for data of the *reliable* actuator traffic, whereas data of the sensor traffic is more likely to be evicted. Another expected observation is that *probabilistic* caching further improves the cache hit ratio thanks to its increased CS diversity.

5.5 Preventing Starvation of the Commons

5.5.1 The Problem of Resource Starvation

Given that content in the CS is only replaced if new content arrives and that unprioritized content may not replace prioritized content according to the rules laid out in Section 5.2.2, it is easy to construct a scenario in which all CS space is occupied by prioritized content, thus starving the unprioritized. An initially distributed set of prioritized data chunks, for example, could block caches for an unlimited time if followed only by content of lower priority. Moreover, if content request rates reach a frequency at which retransmissions become necessary in order to fulfill the requests, the fact that unprioritized content cannot rely on the CS as a retransmit buffer means that unprioritized flows may no longer be delivered at all.

One might expect the PIT to be affected by starvation in a similar manner, with *prompt* entries crowding out *regular* entries; in practice, however, this does not pose a problem, since as we have shown in Sections 5.3.4 and 5.4.2, the improved PIT coordination gained through QoS coordination leads to a more efficient utilization of the PIT space on the whole, while at the same time PIT entries are erased quickly or time out.

5.5.2 Countermeasures and Experimental Evaluation

We can prevent starvation of CS utility for unprioritized content by introducing fairness measures that prevent prioritized content from blocking the CS and can guarantee that some non-zero amount of unprioritized content can always be cached.

Countermeasure for Cache Blocking. The first countermeasure is the introduction of a priority decay time τ. Any prioritized content that has been in the CS for a time equal to or larger than τ is reclassified as unprioritized and may thus be replaced by unprioritized content. In practice, τ may be chosen as an average time of cache utility. The firing of this timer will prevent prioritized content from blocking the cache for longer than useful in a mixed environment.

Countermeasure for Cache Squeeze Out. The previous measure alone, however, may not be sufficient if the rate of incoming prioritized content is high, as the replacement rules defined in Section 5.2.2 state that unprioritized content is always replaced before prioritized content, meaning that the actual time the unprioritized content is allowed to stay in the CS may be too short to be useful.

There are many ways to counter this threat of squeezing unprioritized content completely out of the caches, such as preallocated cache resources and modified timers. Most of these countermeasures depend on parameters specific to the traffic patterns and are therefore difficult to deploy in a general way. We argue for a simple, generic solution as given by probabilistic caching, which as we recall from Section 5.4 also enhances cache diversity. We will now quantify the effects of probabilistic caching in a starvation scenario.

Experiment Configuration. We want to measure resource utilizations of the PIT and CS data structures in setups with increased levels of network stress that are prone to starvation. As before, we limit the PIT and CS resources on every node in the topology to hold a maximum of 5 elements. We consider Scenario 2 as described above, but reverse the priority roles: Sensor

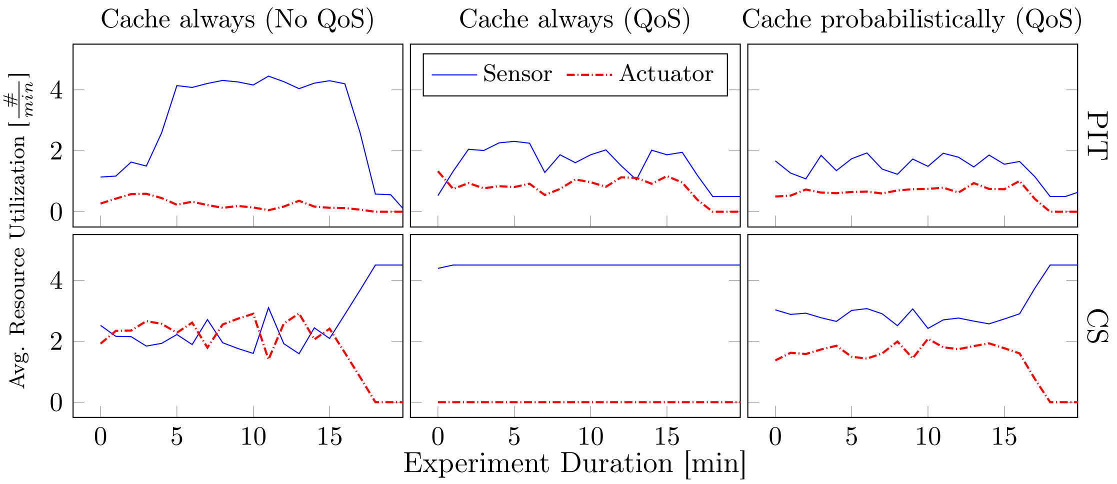


Figure 5.13: Evolution of resource utilizations for a successor node of the gateway and an actuator request interval of 5±1 s.

traffic is now prioritized, while the cache-dependent actuator traffic remains regular. In this setting, we thus emphasize the effects of cache starvation for unprioritized data traffic.

Resource Utilization. In this experiment, we measure the resource utilization over time in order to identify starved traffic flows in environments with limited PIT and CS resources and increasing network loads. Figure 5.13 depicts the evolution of resources from the point of view of node g_{sr}, which is a successor to the gateway and resides on the right wing of Figure 5.5. g_{sr} is root of a subtree with 18 descendants and thus will experience extensive traffic in both directions. Actuators emit unprioritized request-response flows in groups and the gateway transmits prioritized sensor readings.

We first observe the distribution of sensor- and actuator-bound PIT entries. Without prioritization, the PIT is mostly saturated with gateway traffic and repeatedly denies actuator requests. This results from the close proximity of g_{sr} to the gateway node and the concomitant fast placement of gateway requests into the PIT. Consistent with our observations in the previous sections, PIT exhaustion relaxes with QoS enabled and resource occupancies almost equal out between the traffic types due to better PIT coordination and faster consumption of pending Interests. This improves slightly further with probabilistic caching in place, since effectiveness increases and more content can be served from caches. In both cases with QoS prioritization enabled, no signs of starvation are visible at the PIT level.

In contrast, the occupation of the Content Store clearly shows how unprioritized content gets blocked from caching for the priority-guided cache always policy. QoS features hence lead to a straight starvation of the CS resource for unprioritized actor traffic. Probabilistic caching—used with the previously established probabilities 0.7 for prioritized and 0.3 for regular traffic—

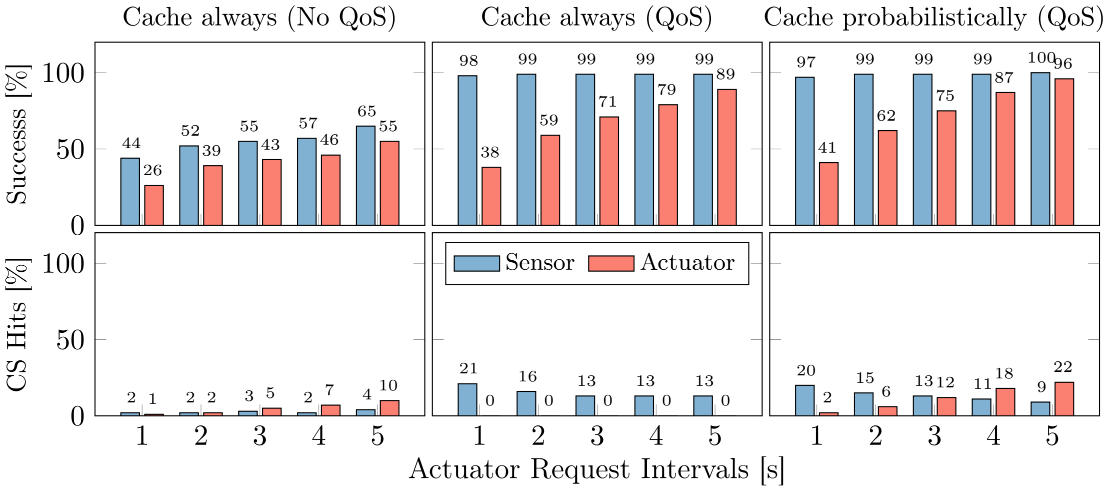


Figure 5.14: Average success rates and cache hits for different levels of actuator request intervals.

seamlessly resolves this starvation and enables about 30% cache placement for regular packets on average.

Success of Network Operations. Figure 5.14 displays the corresponding success rates for packet deliveries and cache hits for a series of actuator request intervals. Network degradation and cache underutilization are clearly visible in the absence of QoS as a result of uncoordinated network resources. Activating QoS with the cache-always policy significantly improves the packet delivery success for both traffic classes (as observed before), but prevents cache hits due to the cache starvation observed above.

In contrast, probabilistic caching maintains the utility of the caches for the unprioritized actuator traffic. Moreover, it improves the success rates of actuator traffic without sacrificing performance of privileged communication.

5.6 Showcase: QoS Impact in Disaster Scenarios

We now demonstrate how differentiated ICN resource management can serve the needs of challenged deployments such as constrained IoT edge networks in disaster scenarios. Using realistic implementations on RIOT, we demonstrate how very constrained devices in harsh environments can reliably communicate, provided QoS measures are in place. These devices gradually invoke traffic flows of different priority levels. In this setup, we contrast regular bulk traffic admitting degradation in flow latency and reliability with QoS-enhanced traffic differentiation and visualize the improved flow resource consumption of high priority traffic on all nodes.

Challenges in disaster scenarios. In unforeseen disasters, a quick response is imperative to prevent serious harm to people or the environment and to minimize collateral damage. With

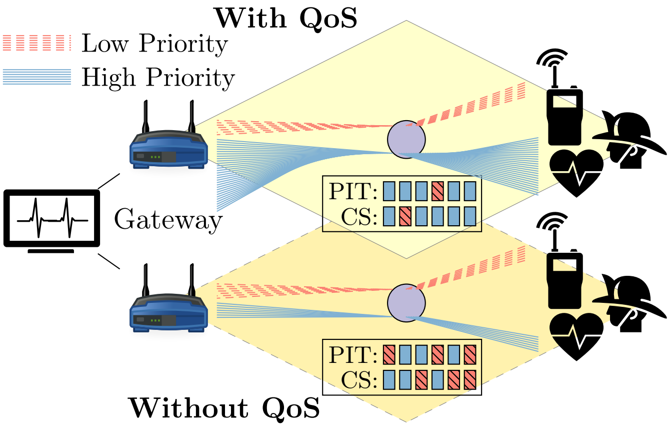


Figure 5.15: Resource allocation with and without QoS mechanisms in a disaster scenario.

this objective, first responders are sent into the field to (*i*) assess the situation on-site, (*ii*) immediately administer first aid, and (*iii*) call for appropriate further emergency support.

For an efficient operation, search and rescue forces require operational guidance by command and control centers. Since regular communication infrastructures are often not available, first aiders are forced to deploy spontaneous radio networks to enable data flows of different importance under intermittent connectivity.

Typically, hand-held communication devices attach to field units and report back readings from sensors that are deployed in the equipment or the immediate vicinity. Sensor devices are battery-operated with a small form factor in order not to interfere with the mobility and maneuverability of rescuers. NDN deployments have shown great potential to successfully operate in disaster scenarios [158] by inherently providing in-network caches and intrinsic multicast capability as well as support of consumer mobility.

Networked sensor devices form an ad hoc low power lossy network (LLN), where device limitations can significantly impact the overall network performance. To mitigate performance degradations for continuous and life-critical traffic flows such as ECG heartbeat monitoring, the network must be able to differentiate between traffic flows, become aware of flow-specific priorities, and balance available resources according to these priorities.

The rescue case. Figure 5.15 illustrates our general setup for the two configurations, with and without QoS features enabled. Our gateway continuously requests heartbeat sensor readings and displays them on a dashboard. In addition, our communication device polls new audio messages every 10 seconds to receive up to date fireground assistance. Once an audio message exists, the message is bulk requested, thereby leading to resource saturation on the forwarder and a disruption of continuous heartbeat readings on the gateway. We repeat this setup with and without QoS features enabled and assign a high prioritization (prompt) to the heartbeat flow. On the dashboard, we illustrate an uninterrupted heartbeat signal to indicate the low latency and high reliability characteristics of our prioritized traffic flow, while slightly distorted audio

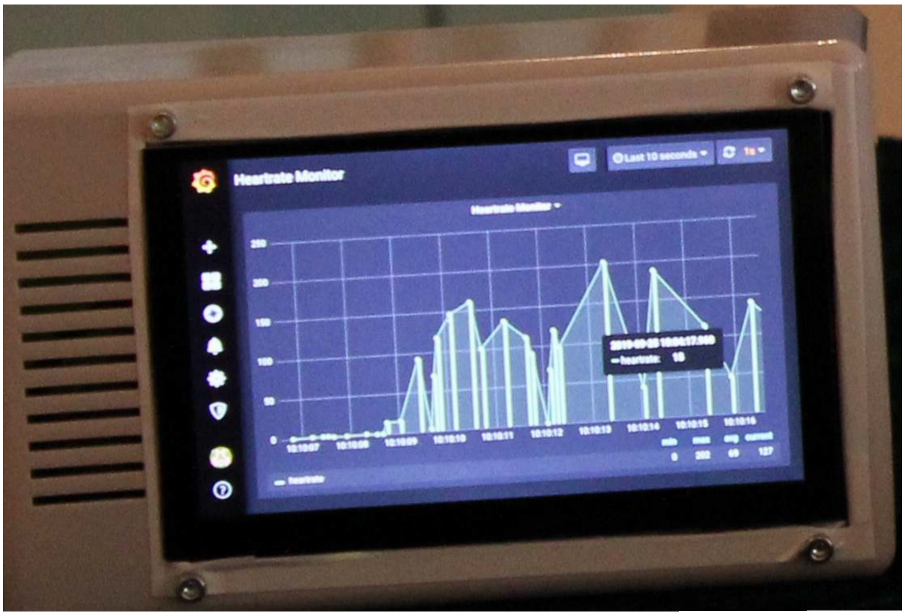


Figure 5.16: Heartbeat signals with and without QoS mechanisms.

messages are still being dispatched to our communication device on the fire ground. Figure 5.16 pictures the results: While heartbeats without QoS support barely arrive at the gateway, we see strong signals after the QoS features are enabled.

5.7 Conclusions

We presented and analyzed QoS extensions to NDN that are suitable for constrained devices. Starting from a name-oriented flow classification scheme, we introduced the two service dimensions *prompt* and *reliable* network forwarding. Strategies were defined that not only foster local, isolated resource allocations, but take into account coordinative actions between different internal resources of a node, as well as correlations between nodes. Here, we exploited the rich set of forwarding and caching options that NDN includes, while protecting resources from starvation by applying fairness measures.

We were able to validate our approach in real-world experiments on a large testbed using a realistic multi-hop wireless setup and in a realistic use case implementation for the rescue domain. Moreover, we learned that QoS management in NDN is not confined to simple resource trading, but can lead to a global enhancement of network performance by optimizing the interplay among various resource consumptions. In particular, we found evidence that (*i*) coordination of PIT and CS has a prevailing effect on the overall performance of the networked system, and (*ii*) incorporation of Interests in QoS treatment is vital to cater for resource coordination.

Chapter 6

Resilience in Harsh and Mobile Networks for ICN

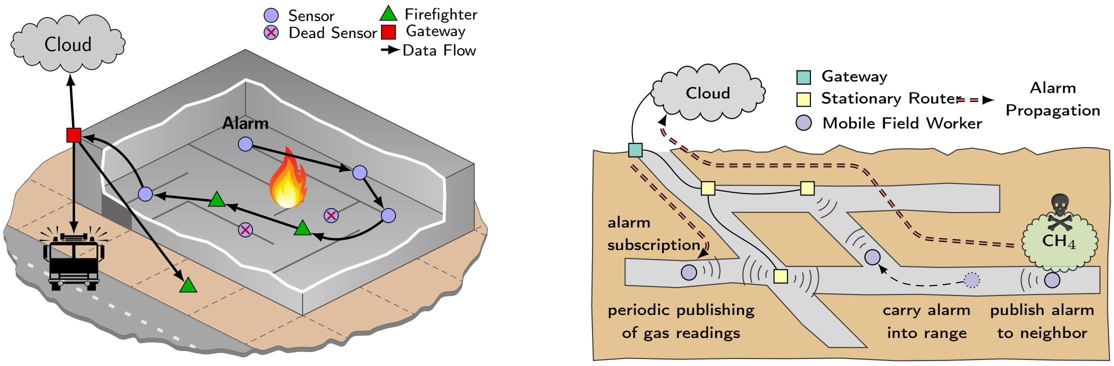
Abstract

The Information centric networking paradigm has proven particularly useful for the constrained Internet of Things (IoT), in which nodes are challenged by end-to-end communication without network assistance. This work focuses on the interaction between possibly mobile sensors *and actuators* in such IoT regimes which deploy the Named-Data Networking (NDN) architecture. Constrained nodes in interactive scenarios need to be highly responsive but can only manage limited control state. We argue that the request-driven NDN networking paradigm, which prevents pushing of unsolicited data, should be preserved to confine the attack surface, whereas unsolicited *link-local* signaling can accelerate responses without sacrificing security.

In this chapter, we contribute HoP-and-Pull (HoPP), a robust publish-subscribe scheme for typical IoT scenarios that targets low-power and lossy wireless networks running hundreds of resource constrained devices at intermittent connectivity. Our approach limits in-memory forwarding state to a minimum and naturally supports producer mobility, temporary partitioning of networks, data aggregation on intermediate hops, and near real-time reactivity. We thoroughly evaluate the protocol by experiments in a realistic, large testbed with varying numbers of constrained devices, each interconnected via IEEE 802.15.4 wireless LoWPANs. We compare HoPP with common ICN pub-sub and mobility schemes as well as with basic MIPv6 and anchor-based multicast mobility. Implementations are built on CCN-lite with RIOT and support experiments using various single- and multi-hop scenarios.

6.1 IoT Use Cases

In this section, we focus on two use cases for the deployment of mobile IoT devices in industrial facilities and for safety control in harsh environments.



(a) Resilient machine-to-machine communication for the Industrial IoT with heterogeneous constrained IoT devices.

(b) IoT use cases in industrial settings with device mobility and network partitionings.

Figure 6.1: IoT use cases with mobile devices and intermittent connectivity.

6.1.1 Resilient machine-to-machine communication

Industrial settings like oil rigs and warehouse facilities deploy battery-operated devices for collecting sensory data to meet mission-critical requirements and regulatory compliances. Presence detection, smart lighting control, and tracking of exposure to hazards are instances of essential tasks to ensure a safe workplace by mitigating risks to employees and inventory. Replacing or recharging batteries of IoT devices oftentimes incurs high maintenance expenses, especially in confined or hardly accessible spaces. Long battery lifetimes ranging from years to decades are thus desired to minimize deployment costs and are typically achieved with tailored software platforms enabling an optimal power management [159].

Quasi-stationary infrastructures connect these resource-constrained devices to powerful gateways and cloud services using wireless low-power and lossy networks (LLNs) [160]. Notably for regimes with a multitude of IoT endpoints in a single wireless broadcast domain, spuriously disappearing links and saturating network resources are common characteristics. In addition to an energy aware device duty cycling [161], these limitations pose challenges for the host-centric approach to networking, which performs best with perpetual connectivity.

We revisit this basic use case in Figure 6.1a, where a timely reporting of incidental threats to industrial settings is decisive for on-site personnel. In this particular scenario the infrastructure is damaged by a fire outbreak, which leaves the network in a partitioned state. Fire fighters and first responders reconnect the network with hand-held devices to receive crucial data on the whereabouts of trapped or unconscious staff members. Seamlessly handling heterogeneous devices and coping with intermittent infrastructure loss are significant qualities of the network in these settings.

6.1.2 Industrial safety networks

Industrial safety and control systems are increasingly interconnected and operate under harsh conditions. In this use case, we consider industrial environments with a threat of hazardous contaminant (*e.g.,* explosive gas) that need continuous monitoring by stationary, as well as mobile sensors like depicted in Figure 6.1b. In case of an emergency, immediate actions are required such as issuing local alarms, activating protective shut-downs (*e.g.,* closing valves, halting pumps), initiating a remote recording for first responders and forensic purposes.

Typical industrial plants are widespread with sparse network coverage, so that mobile workers or machines face intermittent connectivity at scattered gateways. Some sensors and actuators are infrastructure bound, others are independent and battery-powered (*e.g.,* body equipment). The latter resembles the challenges faced in previous DTN-work such as in mines [162].

Like the previous, this use case relies on a fast sensor-actuator network including embedded IoT nodes. In addition, the harsh industrial environment raises the challenges of mobile, intermittently connected end nodes, and network partitioning. Still, enhanced reliability is required in the safety context. We will show in this work, how configurable data replication with dynamically generated content proxies can meet these challenges and how they combine in a lightweight system suitable for real-world deployment [163].

6.2 The Problem of Information Centric IoT Networking and Related Work

6.2.1 Device mobility and network disruptions

Mobile nodes are part of many IoT deployments. While mobility is natively supported at the receiver side of NDN, publisher mobility is considered difficult to solve in a generic way [164]. Translated to IoT use cases, this means mobile sensors are hard to integrate—a particular problem for surveillance and safety sensing applications [163]. Stationary devices can also face mobility issues due to coverage changes in a wireless topology. Related scenarios may also experience temporary network partitioning, which can be treated with correspondence to network mobility.

KITE [165] takes a soft-state approach where mobile producer nodes proactively build temporary paths (*traces* [166]) to a rendezvous point as soon as they move or anticipate data retrievals from consumers. Consumer traffic generally traverses the rendezvous servers, but the hop-by-hop forwarding of NDN can reduce path stretch for consumers that share parts of the path with a mobile producer.

An alternative suggestion that handles an anchor-less producer mobility is provided by *MAP-Me* [167]. In this extension, mobile producers send special Interests to name prefixes they own in order to update obsolete forwarding states. With the premise that an underlying routing protocol operates much slower, such Interests traverse to the old locations of recently moved producers. Any recipient of a special Interest then updates its forwarding information base by recording the incoming face alongside the name prefix inside the Interest.

A recent publish-subscribe system with consumer and producer mobility support [168] uses persistent PIT entries to serve multiple Data packets along a request path. A practical cleanup routine reveals the absence of expected data (*e.g.,* due to mobility) and tears down unnecessary soft-state in the network. Data packets carry infrastructure-specific information to detect routing inconsistencies and to trigger routing repairs.

Other approaches separate human-readable content object names and network address locators in order to handle producer mobility—a concept that is part of the MobilityFirst [169] design considerations and is also adopted for NDN based architectures [170, 171, 172]. Solutions that use a controlled flooding of Interests and broadcast mechanisms of the link-layer [173, 174, 175] show less infrastructure requirements, but generally produce more signaling overhead.

Systems that act on the information-centric maxim already exhibit a decoupling of data and location and therefore potentially qualify for deployments with long network disruptions. Delay tolerance for NDN can be enhanced by integrating the Bundle protocol [176], or with specific content caching mechanisms [177].

6.2.2 Naming and routing

Naming content on an information-centric network layer promises a simplified access to information. Routing on names directly designs a lean network without further address mapping. It obsoletes infrastructure like the DNS and eliminates the attack surface inherent to the mapping. Both aspects are of great advantage in a constrained IoT network. However, name-based routing encounters the problems of (*i*) exploding routing tables, as the number of names largely exceeds common routing resources, and (*ii*) limited aggregation potentials, as names are specific to appliances and applications, but independent of content locations. More severely and in contrast to IP, a local router cannot decide on aggregating names since the symbol space of names is not enumerable in practice [44]. Limiting the complexity of name-based routing and FIB table state is one of the major challenges in IoT networks [126].

Routing normally proceeds according to location information from the FIB. Names in FIBs only aggregate well if naming follows the topological structure of the network. This rarely holds, since naming is application-specific, and cannot be detected without distributed knowledge. To overcome FIB explosion, several authors refer to the NDN capabilities of stateful forwarding, using the option of distributing requests to several interfaces simultaneously [178, 179]. Such Interest multicasting will lead to duplicate content deliveries if the network is densely connected. In 'Pro Diluvian' [180], the effects of such scoped flooding are analyzed, and authors find a utility limited over very few (≈ 2–3) hops. Such opportunistic forwarding can also lead to loops, as was pointed out by Garcia-Luna-Aceves [181]. In any case, the excessive traffic, as well as redundant PIT states make this approach infeasible for the IoT.

COPSS [41], an earlier publish-subscribe approach inspired by PIM [182] multicast routing, selects a rendezvous point to interconnect publishers and subscribers. Such dedicated routing point naturally allows for name aggregation. Like PIM-SM (Phase 2), COPSS further establishes a dedicated forwarding infrastructure (subscription table) that establishes persistent forwarding paths from the publisher via the rendezvous point to the receivers.

A publish-subscribe framework with a focus on building management systems (ndnBMS-PS [183]) uses the functionality of repositories to publish data following signed command Interests from producer nodes. Repositories then replicate in the network and content synchronizes to subscriber nodes using a synchronization protocol for NDN. In contrast to ndnBMS-PS, which requires an external topology management, the publish-subscribe Internet (PSI) [184] architecture provides the two network primitives publish and subscribe, as well as topology managers for path computations between host nodes. Similar to COPSS, PSI uses rendezvous points to match announcements and subscriptions. TPS-CCN [185] is a topic-based publish-subscribe CCN system that integrates a MANET link state routing protocol to identify available topics in the network. The naming scheme includes the topic prefix and appends a version number to track the evolution of content for a specific topic. Subscribers then request content objects using standard Interest messages for names with progressively increasing version numbers. In case of network disruptions, a delay-tolerant mode allows for broadcasting Interests to explore the close vicinity for desired content.

MFT-PubSub [186] builds a spanning tree on an IP network overlay using a leader election algorithm. Subscriptions propagate along the tree topology to all brokers and are recorded in the corresponding routing tables. Announcements are then forwarded to subscribers according to the existing forwarding state. On network partitioning, local leaders are elected to maintain the routing infrastructure in each partition. Other approaches either organize topics under common prefix trees [187, 188] to rely on the prefix matching capabilities performed by the NDN forwarding fabric, or directly utilize Interest messages to push data towards subscribers [189].

PANINI [44] re-uses the idea of an aggregation point called Name Collector, but does not establish a (persistent) forwarding plane like COPSS. Instead, PANINI uses selective broadcasts to discover unpopular routes towards the network edge. For the IoT, we want to minimize control traffic and avoid flooding. Therefore, we restrict our solution to a lean default routing, instead.

6.2.3 ICN in the IoT

It became apparent [190, 110, 111] that ICN/NDN exhibit great potentials for the IoT. Not only allows the access of named content instead of distant nodes a much leaner and more robust implementation of a network layer, but in particular prevents the request-response pattern of NDN any overloading with data at the receiver.

Security, resilience, and robustness

For a few years, it was the belief that NDN can be DoS resistant by design, until Interest- and state-based attacks were discovered [191]. Subsequent work [192, 127] elaborated the threats of Interest flooding and overloading FIB and PIT tables by user-generated names and content requests. This has proven difficult to mitigate [193] and is a particular threat to memory-constrained nodes. In the subsequent Section 6.3, we will show how a FIB with simple default routes can serve the IoT, and how PITs remain minimal by hop-wise content replication between nodes.

ICN deployment in the IoT has been studied with increasing intensity, touching protocol design aspects [194, 98, 31], architecture work [195, 99, 39], and practical use cases [102, 103, 196, 163]. Emerging link-layer extensions for the wireless like TSCH turned out to be beneficial for the interaction of NDN communication patterns and channel management [197]. Several implementations have become available. CCN-Lite [106] runs on RIOT [28] and on Contiki [198], NDN has been ported to RIOT [108]. Thus, grounds seem to be prepared for opening the floor to real-world IoT applications with NDN.

Many deployments in the IoT, though, follow the communication patterns *on demand*, *scheduled*, and *unscheduled*. Actuators in particular rely on unscheduled control messages. Since NDN is built on the request-response scheme of data-follows-Interest, unscheduled push messages are not natively supported. For the IoT, this has been identified as a major research challenge [126].

Push communication

Several extensions have been proposed to enable an unsolicited push of data, among them *Interest-follows-Interest* [102], *Interest notification* [199], and a dedicated *push packet* [200]. All these push messages are sent immediately to a prospective consumer node, which not only conflicts with the ICN paradigm of naming content instead of hosts, but has no forwarding supported on the network layer. No push packet will reach its destination unless potential receivers are announced to the routing using a node-centric name. Unidirectional data push to named nodes, however, lacks flow as well as congestion control, and opens an attack surface to DoS. In the IoT with its constrained nodes, this constitutes a particularly severe disadvantage.

Carzaniga et al. [40] with a proposal of *long-lived Interest* seem to be the first in addressing the push challenge in a natural NDN fashion. Subscribers issue a persistent Interest that is not consumed at content arrival, and thereby establish a (static) data path from the producer. Unfortunately, long-lived Interests open an unrestricted data path to the recipient and thereby inherit the threats of overload as other push primitives. In addition, persistent forwarding states in PITs lead to self-reinforcing broadcast storms whenever L2 broadcasts are used [109]. Finally, frequent topology changes as characteristic for the IoT will routinely break paths. In the following, we will show how regular Interests with appropriate lifetime can serve this purpose equally well, without suffering from its drawbacks.

Category	Characteristics	References
Routing and mobility	Use cases & surveys	[164, 166, 126]
	Producer mobility	[165, 168, 167]
	Locator / identifier split	[169, 170, 171, 172]
	Delay-tolerance	[176, 177]
	Opportunistic routing & signaling	[173, 174, 175, 180]
	Stateful & adaptive forwarding	[178, 179, 44, 181]
Publish-Subscribe	Rendezvous-based delivery	[41, 182, 184, 186, 189]
	Dataset synchronization	[183]
	Semantic naming & routing	[187, 188, 185]
Information-centric Internet of Things	Architectures & deployment reports	[163, 190, 110, 111, 194, 98, 195, 99, 39]
	Security threats & analyses	[191, 192, 127, 193]
	Link-layer extensions	[197, 31, 109]
	Unsolicited data delivery	[102, 199, 200, 40]

Table 6.1: Overview of the related work grouped according to the main contributions.

6.2.4 Requirements for an ICN-based IoT Publish-Subscribe system

Typical IoT scenarios impose critical requirements on a publish-subscribe system. The state of the art as summarized in Table 6.1 mainly focuses on general purpose deployments with sufficiently provisioned network and device resources. We derive three challenges from the related work, which need to be addressed for creating a robust and energy frugal content replication mechanism. First, IoT deployments may consist of hundreds of resource constrained devices with intermittent connectivity. A publish-subscribe approach for these setups needs to minimize forwarding states as they scale with the number of network participants. While a powerful gateway device can potentially hold enough in-memory forwarding information to represent the whole network, the constrained devices have rather limited memory space for forwarding states. Second, consumer and producer mobility as well as temporary network partitionings are prevalent and must be handled by any publish-subscribe solution for the IoT. Regular packet loss is expected and corrective actions without an excessive overhead is required, while not inhibiting the protocol reactivity. Third, the constrained processing and memory resources of common IoT hardware necessitate a low implementation complexity. All applications on an IoT software platform compete for available resources and wasteful uses limit the device operability.

In Section 6.3, we design a robust publish-subscribe system that meets these requirements to enable a resilient content replication. A real protocol implementation for an IoT operating system demonstrates the feasibility of the described approach.

6.3 HoP and Pull: A Publish-Subscribe Approach to Lightweight Routing on Names

6.3.1 Overview

We now describe HoP-and-Pull (HoPP), our pub-sub system for lightweight IoT deployment. For a confined IoT environment, we make the common assumption that nodes form a stub network that may be connected to the outside by one or several gateways. Some global prefix is given to a gateway, but (wireless) IoT nodes can reach a gateway without global prefix changes in one or several hops unless they are temporarily disconnected [68]. Internally, nodes may be grouped according to one or several sub-network prefixes (*e.g.,* `/valves`).

We select one or several distinguished nodes to serve as Content Proxies (CPs) at infrastructure setup time. CPs are typically more stable and more powerful than the constrained edge nodes. The CP function may reside on gateways or other infrastructural entities of a deployed system. These CPs take the role of data caches and persistent access points. They will be reachable throughout the network by default routes, unless temporary partitioning occurs. Note that one CP can serve several local prefixes, but a local prefix may also belong to several CPs. The latter scenario will lead to replicated caching with higher and faster data availability.

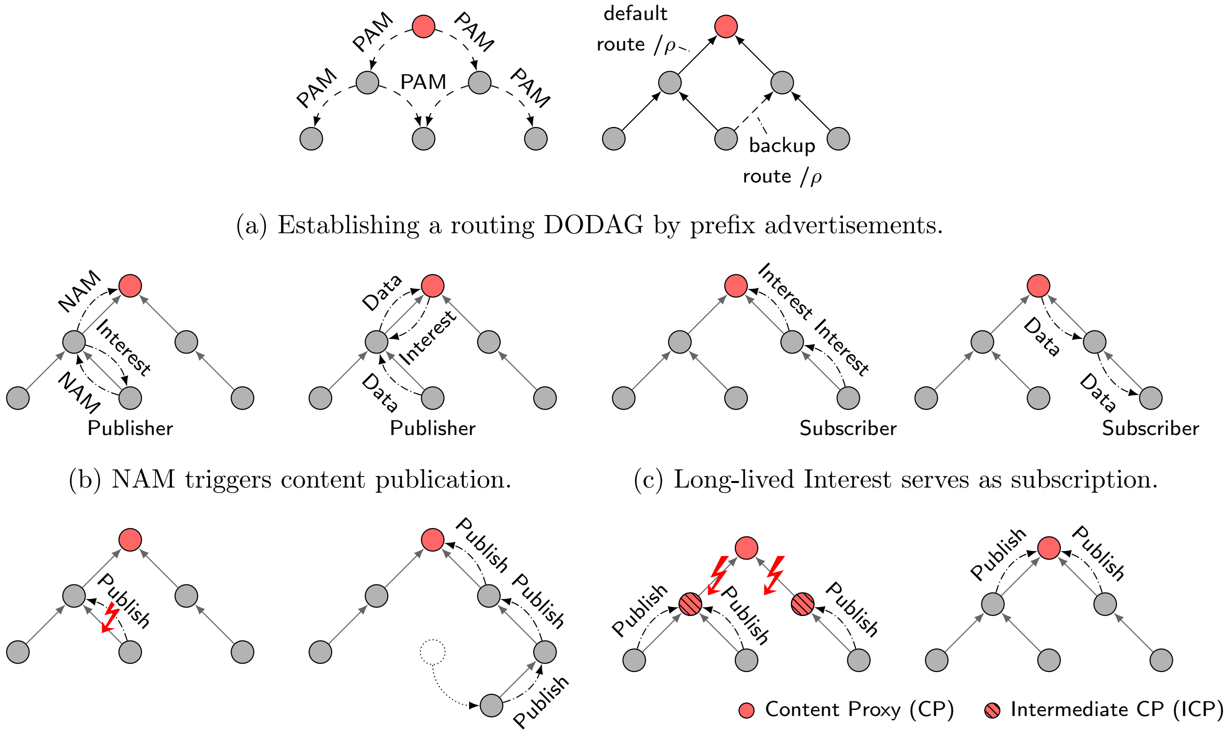
Our publish-subscribe protocol for the IoT is then composed of three core primitives:

1. Establishing and maintaining the routing system
2. Publishing content to the Content Proxies
3. Subscribing content from the Content Proxies

Our following protocol definition strictly complies with the design principles: (*i*) minimal FIBs that only contain default routes, (*ii*) no push primitive or polling, (*iii*) no broadcast or flooding on the data plane. The HoPP protocol transparently manages consumer and *producer mobility* as could be demonstrated in our prototype [201].

6.3.2 Prefix-specific default routing

Content Proxies advertise the prefix(es) they own on the control plane to all direct neighbors in a Prefix Advertisement Message (PAM). PAM messages are link-local, and do not interfere with regular NDN network operations. This orthogonality leaves the primary data structures Pending Interest Table (PIT) and Content Store (CS) unaffected. Hence, route signaling performs a topology discovery on a strictly scoped and shielded control plane. Observing nodes will adopt a CP as their parent and re-distribute the PAM message to their neighbors with an increased distance value. Much like in the core RPL [202], parents broadcast PAMs to the one-hop vicinity, which allows for an increased scalability independent of the neighborhood size. Trickle [203] regulates the rate of message transmissions to substantially reduce the broadcast chattiness in



(a) Establishing a routing DODAG by prefix advertisements.

(b) NAM triggers content publication.

(c) Long-lived Interest serves as subscription.

(d) Rejoining DODAG on publisher mobility.

(e) Interim Content Proxies (ICPs) buffer publishings in partitioned networks.

Figure 6.2: Overview on the HoPP protocol operations: topology management, publish-subscribe, mobility and delay-tolerance support.

stable network topologies. All nodes will become members of a Destination-Oriented Directed Acyclic Graph (DODAG) while routing converges. Any topological change, *e.g.,* due to mobility or parent timeouts, resets the Trickle algorithm to quickly trigger PAM announcements and, thus, converge towards a consistent DODAG with refreshed forwarding states, before reducing the chattiness again.

Nodes select the best seen uplink in their FIB as default route to the announced prefix, but may add additional uplinks with lower priority for backup. The selection process uses the hop-count metric, but also allows for the integration of more sophisticated alternatives, *e.g.,* MRHOF [204].

A deployment always includes at least one CP serving a specific name prefix. If multiple CPs announce the same prefix, then nodes configure multiple default routes for this particular prefix. Unlike in host-centric deployments, NDN inherently supports a multi-destination forwarding and thus enables a seamless data replication onto several CPs.

Figure 6.2a visualizes the PAM prefix distribution and the corresponding FIB entry for the sample prefix $/\rho$. All nodes establish prefix-specific default routes on their shortest paths upstream. In addition, nodes learn backup paths of equal hop distance, which may be of lower radio quality.

6.3.3 Publishing content

An IoT node (sensor) that has new data to publish will first select a name. It may choose either from a predefined scheme accessible by local controllers, some common standard set, or decide individually. Since generated names are expected to be unique across the whole system, they include device-specific identifiers to partition the naming scheme. It is typical for time series sensor data to append an increasing counter or the current timestamp to a name prefix. If the uniqueness of names cannot be guaranteed within the stub network, then a duplicate name detection is necessary. The specific method is out of scope of this document, but the multihop duplicate address detection (DAD) of the neighbor discovery protocol (NDP) [205] provides a viable base.

It will advertise this content name to its upstream neighbor via a (unicast) Name Advertisement Message (NAM). It will also associate the content with one or several topic names and adds these to the content metadata. Depending on the publishing rate of content, a node can announce multiple names in a single NAM message. This aggregation of names is however limited by the maximum transmission unit (MTU) of the underlying link-layer.

Under regular network conditions, the upstream neighbor is expected to retrieve the advertised content via the incoming interface of the NAM. It proceeds according to the standard NDN scheme: An Interest requests the name, the data is returned in response. Concurrently, the upstream issues a corresponding NAM to its parent, which in turn pulls the content one hop closer to the CP. This hop-wise content replication proceeds until the data arrives at the CP.

It is worth noting that the NAM content alerting is situated on the control plane using *link-local unicast* signaling. Neither a data path is established in the PIT, nor are FIBs modified. NAM content signaling also complies with the strictly scoped and shielded control plane of HoPP.

The publishing mechanism is depicted in Figure 6.2b. A Publisher issues a NAM to its parent, which requests the content and republishes the NAM towards the CP in parallel. The content request is performed on the NDN data plane via a regular interest-data handshake. If a single NAM includes multiple name announcements, then each of the name is requested separately. After arrival of the data, nodes satisfy outstanding Interests up to the CP.

During irregular network conditions, a node may not receive an Interest that matches its previous name advertisements. This may be due to broken links, failing or deep-sleeping nodes, or enduring overload. After a deployment-specific timeout, the content owner will adapt and try to publish the content on an alternate path by sending a NAM up on a backup link. In case of a complete failure, the content node can follow two strategies: Either it waits and re-advertises according to an exponential back-off, or it solicits a refresh of router advertisements for learning new, operational routes. In the latter case, nodes send multicast SOL (solicitation) messages to trigger PAM messages from immediate neighbors.

6.3.4 Subscribing to content

A subscriber in HoPP behaves almost like any content requester in NDN. It issues a regular Interest request up the default route to the CP and awaits the response. There are two deviations from plain NDN, though. First, the subscriber cannot extract content names from its FIB, since FIBs only contain prefixes. These prefixes, however, can serve as topics in the context of confined application deployment. Second, to meet real-time alerting requirements of publishers, a subscriber can issue timely Interests with extended lifetimes to immediately receive published content once it arrives at the content proxy.

Names are expected to follow an application-specific logic. In a publish-subscribe system, individual names of content items are grouped according to topics, which itself appear as prefixes in the naming hierarchy. The corresponding CP will answer the request for a topic with an empty data chunk that carries available content name(s) as metadata, *e.g.,* in a manifest [206].

Figure 6.2c displays the operations of a subscriber. An Interest for named content is sent up to the proper prefix owner (CP) and remains for a predefined lifetime, if the Content Proxy cannot supply the data. These requests terminate at the CP anchor and do not propagate downwards to the actual publishers. In case content is arriving from a publisher to the CP, data is transferred automatically down the reverse Interest path—as a regular NDN operation. We anticipate that in common sensor-actuator networks of the IoT, the application semantic will define meaningful Interest lifetimes. Otherwise, in regimes of largely fluctuating temporal behaviors or long-lasting subscriptions (*e.g.,* alerts), the subscriber may refresh and maintain the request at its discretion.

Note that in contrast to *long-lived Interests* or the COPSS *subscription tables*, such Interests of extended lifetime are consumed by arriving content and do not open a persistent, uncontrolled data path. Subscribers continue to apply flow control and may discontinue subscriptions to unwanted content.

6.3.5 Publisher mobility and network partitioning

A publishing node that moves from one point of attachment to another within the IoT domain, will experience stable routing conditions in the sense that default routes to active prefixes should exist everywhere in a connected network. Correspondingly, the mobile node (MN) can re-configure its upstream route either by waiting for the next prefix advertisement (PAM), or may actively solicit an additional PAM to discover new, reachable parents. Note that these link-local route configurations are of low complexity and closely resemble the autoconfiguration of IPv6 default gateways. In contrast to Mobile IPv6 [207], though, the MN in our publish-subscribe system can continue publication immediately after a link-local route is re-established by a newly arriving PAM as outlined in Section 6.3.2 for building and maintaining the topology. This makes the handover process lightweight and very fast.

Figure 6.2d illustrates provider mobility. A publisher removes from the network while trying to publish a content item and enters the radio range of another node in the DODAG. It may now actively learn about network re-attachment (*e.g.,* from link triggers), or learn from a newly arriving PAM. After the local upstream is configured, the mobile publisher can successfully complete its publishing handshake.

Temporary network partitioning proceeds very similar to mobility. An intermediate node that looses upstream connectivity will explore alternate paths (*c.f.,* Sec. 6.3.3), but has to await a re-attachment in case of a complete failure. Such node will continue to receive publishing demands (NAMs) from the downstream, which it will satisfy in accordance with its resources. On overload, it will terminate to retrieve content from its children. Proceeding this way will establish a classic backpressure mechanism of flow control.

Operations under network partitioning are shown in Figure 6.2e. Following an outage of the CP, immediate children experience a disconnect. Forwarders act automatically as Interim Content Proxies (ICPs) once they lose upstream connectivity. ICPs are temporary content proxies, and they store all published content as long as they have enough buffer resources. They continue to handle publications (as well as subscriptions) until connectivity to the CP is re-established, in which case a forwarder re-publishes all delayed data to the newly chosen upstream parent.

6.4 Implementation and Evaluation

6.4.1 Implementation for CCN-lite on RIOT

We implemented the HoPP extensions on the CCN-lite version ported to RIOT and deploy NDN. It is noteworthy that this software stack supports both, the NDN core protocol as well as CCNx. On RIOT, CCN-lite implements the `netdev` interface and runs as a dedicated single-threaded network stack.

The architecture of HoPP is depicted in Figure 6.3. It mainly adds a new control protocol block that handles exchange and processing of the two new packet types (PAM, NAM) on the control plane. This extends the `forwarder` module of CCN-lite. The `forwarder` allows extensions for the packet parsing by the use of user-defined callback functions on a suite basis. Considering this loose coupling, the actual topology maintenance was implemented separately from the CCN-lite core. The `topology manager` handles PAM scheduling and parent selection to form and maintain the routing topology (DODAG). Resulting forwarding states are reflected in the FIB with the help of the CCN-lite API. The Name Advertisement Daemon (`NAD`) module handles parsing and scheduling of NAM messages. A NAM Cache (`NC`) is used to intermittently track the hop-wise propagation and to reschedule NAM transmissions in case of network disrup-

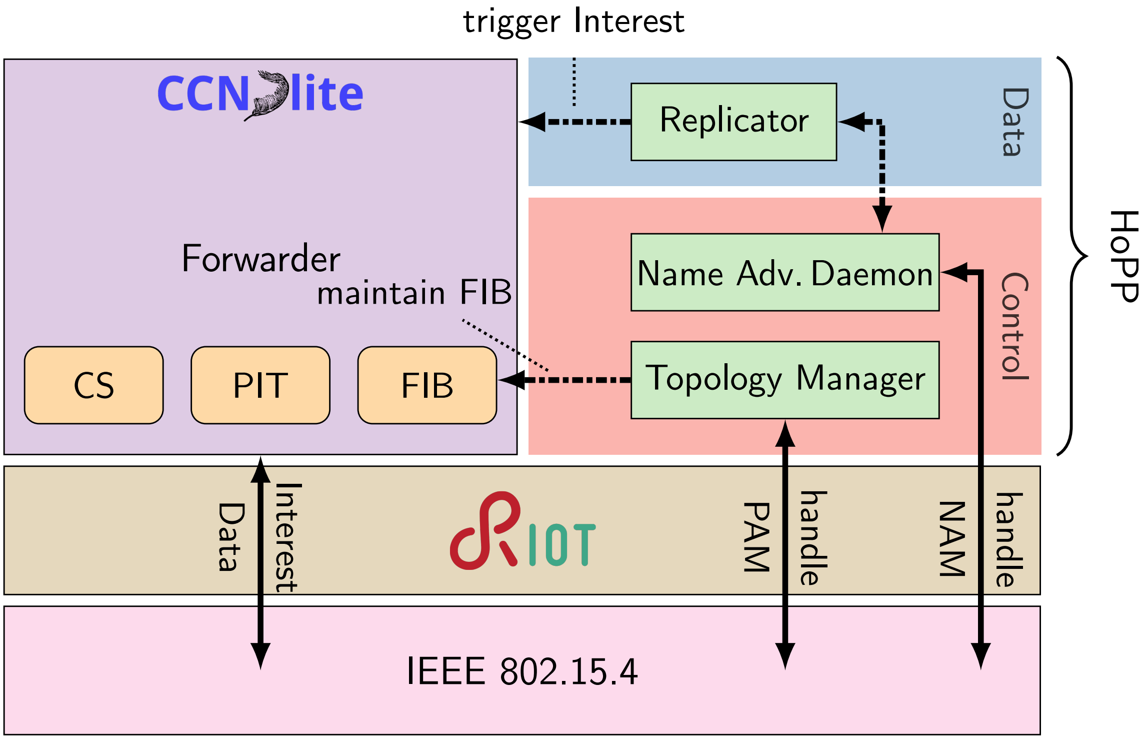


Figure 6.3: IoT Publish-Subscribe architecture.

tions. For each entry in the NC, the NAD triggers the replicator to invoke a hop-wise content replication on the data plane via pull-driven Interest-Data. To ensure hop-wise replication of published content, a caching strategy was added to CCN-lite that hinders replicated content from being cached out while the replication to a parent node is in process. After a successful Interest-Data exchange, the replicator notifies the NAD module and the appropriate NC entry is freed for removal. We note that the newly added data structure (NC) is very lightweight, since it only tracks the names of unpublished content objects. The content itself resides in the default CCN-lite Content Store (CS). The NAM Cache is orthogonal to the other data structures and is implemented outside of CCN-lite within the NAD. HoPP maintains the CCN-lite FIB data structure, but does not interact with the Pending Interest Table (PIT) and CS. The latter structures are regularly updated by the normal NDN operation through Interest and Data messages.

6.4.2 Experiment setup description

All experiments are conducted in the FIT IoT-LAB testbed [27] to reflect common IoT properties. The testbed consists of several hundreds of class 2 [1] devices equipped with an ARM Cortex-M3 MCU, 64 kB of RAM and 512 kB of ROM, and an IEEE 802.15.4 radio (Atmel AT86RF231). The radio card provides basic MAC layer functions implemented in hardware, such as ACK handling, retransmissions, and CSMA/CA. The software platform is based on RIOT [28] and an extended CCN-lite network stack.

The performance of the HoPP publish-subscribe IoT system is evaluated on three different topologies:

Paris is a densely connected topology of 69 nodes all within radio reach.

Grenoble (ring) is formed of a closed rectangle with two double-stacked edges. 178 nodes form a heterogeneously meshed network with a maximal hop distance of four.

Grenoble consists of about 350 nodes, where half of them is situated on the rectangle, the other half forms linear extensions leading outwards. This network supports complex topologies with a node distance up to 9 hops.

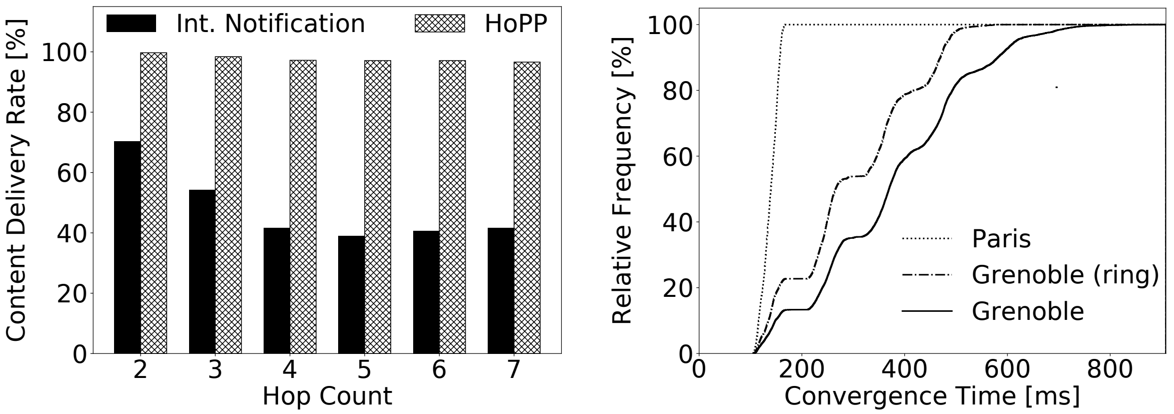
While the physical location of each stationary node is fixed by the actual testbed infrastructure, we carefully select devices that are sufficiently apart from each other to promote the logical formation of meshed multi-hop topologies. Since all nodes are within broadcast range on the Paris site, they always connect to the HoPP content proxy to form a star topology. On the other hand, the resulting meshed topologies for the Grenoble site have many branches with long path stretches. With these topologies, we model the aspects of typical IoT use cases: star topologies are usually deployed in scenarios where devices stay in proximity of a single base station with uplink connectivity. Remote deployments with mobile handhelds in challenging environments, *e.g.,* in confined spaces as illustrated in Section 6.1, use meshed multi-hop topologies.

We illuminate multiple protocol aspects throughout the following sections using the three selected topologies. In Section 6.4.3, we measure the success rates for publishing content as it is an important metric to gauge the efficacy and scalability of this protocol in lossy environments. To assess the reactivity of the pull-driven HoPP approach in these low-power regimes, we compare it against a push-based protocol. Since HoPP also builds and maintains a logical routing topology, we further measure the routing convergence time for the three selected site formations. Section 6.4.4 provides further insights on the resiliency of protocol operations in partitioned and mobile networks.

6.4.3 Evaluation of the HoPP baseline performance

The first evaluation inspects the reliability of HoPP compared to the plain Interest notification [199] approach, which allows including unsolicited data in request messages. We investigate the content reception rate on a given consumer in the Grenoble ring multi-hop topology using a converge cast traffic pattern, where each device generates sensor readings every 30 ± 15 seconds.

Figure 6.4a compares the reliability of HoPP with the common Interest Notification approach in relation to the hop distance of the consumer. For HoPP, we observe a steady high content delivery rate above 96 % for all hop distances in the topology. NDN Interest Notification



(a) Success rate of content delivery to one consumer as a function of hop count in the Grenoble multi-hop testbed.

(b) Routing convergence time for the testbed topologies.

Figure 6.4: Evaluation of the HoPP baseline performance.

admits significantly lower reliability and shows a decline in transmission with increasing hop distance. While a hop count of 1 yields 70 % packet arrivals, success ratio decreases to 41 % for hop distances of 5 and larger. Next, we investigate performance metrics that relate to the temporal behavior of the protocol. Since deficits of the core protocol, but also different failures of networked elements (radio/link layer, CCN-layer, pub-sub, and node layer) translate into delays due to retransmissions and re-arrangements, times to completion are a key performance indicators. In detail, we study (*i*) routing convergence, (*ii*) times to publish content items, (*iii*) times to publish under network partitioning, and (*iv*) times to issue alerts (from publisher to the subscribers).

Routing convergence times in the three testbeds are displayed in Figure 6.4b. Clearly visible is the dependence on hop counts, each counting for an average delay of ≈ 100 ms—the PAM timer. While Paris is a single-hop network and exhibits a single step in distribution, multiple steps represent hop count multiplicities in the multi-hop cases. No exceptional delays become visible. This is due to the moderate timing of the routing protocol which causes a low network utilization.

For the evaluation of the times needed to publish a content item, we iterate the following. For each topology, a Content Proxy is positioned in the center of the network, while randomly chosen nodes publish a single, individually named chunk to the network. Publication is initiated every 30 ± 15 seconds and depending on the nodes position in the tree, one to several data packets traverse the same sub-paths within very short time frames.

Results for the single-hop network (Paris) are displayed in Figure 6.5. Observing round-trip ping values of ≈ 10 ms, the NAM timer ($\mathtt{nam_t}$) of 125 ± 25 ms, and the CCN-lite processing, a mean time to publish of about 135 ms would be expected. Small fluctuations at $\approx 2 \times \mathtt{nam_t}$

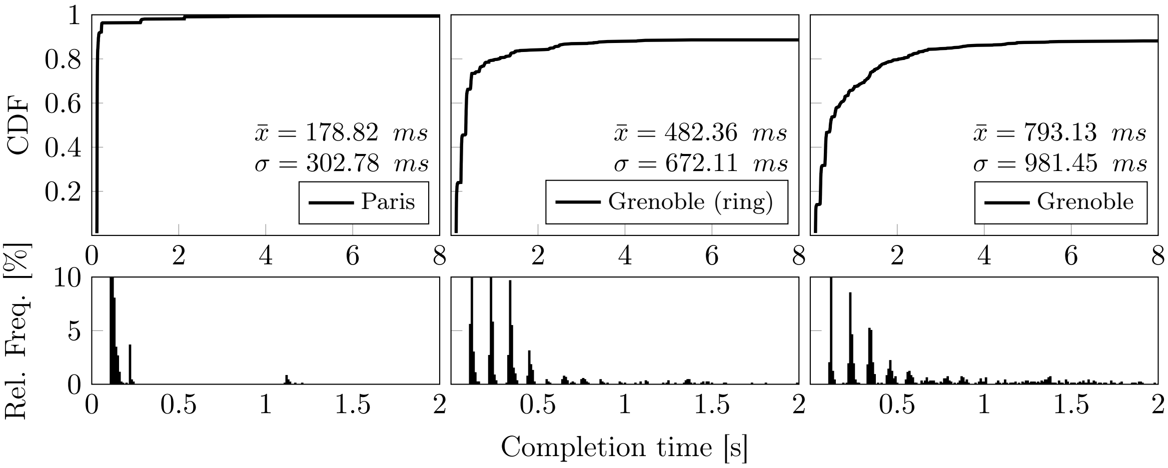


Figure 6.5: Time to content publishing for multiple publisher nodes towards the Content Proxy.

indicate additional delays that result from network disturbances and node congestion leading to paths of hop count two.

Similar results become visible from the Grenoble experiments in Figure 6.5. Clearly pronounced are the first four routing hops, higher hop counts blur according to increasing fluctuations for the Grenoble topology. Roughly 80% of all publish events complete below one second in the Grenoble (ring) topology, while a similar amount of events require up to two seconds for the Grenoble setup. These results clearly show the fragility of the lossy wireless regime, but also confirm a majority of these challenging transmissions did complete on the expected time scale.

Next, the end-to-end delay from the publisher to the subscriber is examined. This corresponds to the use case of issuing alerts between nodes from the local IoT network. In addition to the publish events of the previous measurements in Figure 6.5, this scenario also includes periodic subscription requests that are issued randomly scattered within the topology at intervals of ≈ 30 seconds.

The experimental output for the three topologies are displayed in Figure 6.6. As we might expect, blurring fluctuations have enhanced with only a few pronounced signatures of hops and the means increased slightly by the extended paths towards the subscribers. Notably, the single-hop testbed from Paris performed best under the extended communication load, whereas the full Grenoble testbed clearly runs at its limit. The latter can be easily explained by the many hop transitions required at Grenoble, each of which requires an additional packet exchange which potentially impacts on neighbors within radio range.

Low power lossy networks that connect heavily constrained IoT nodes are known to be incapable of handling such heavy load. We consider it a success that a notable fraction of the content arrived at its receivers on within about 500 ms—a timescale which is considered normal in multi-

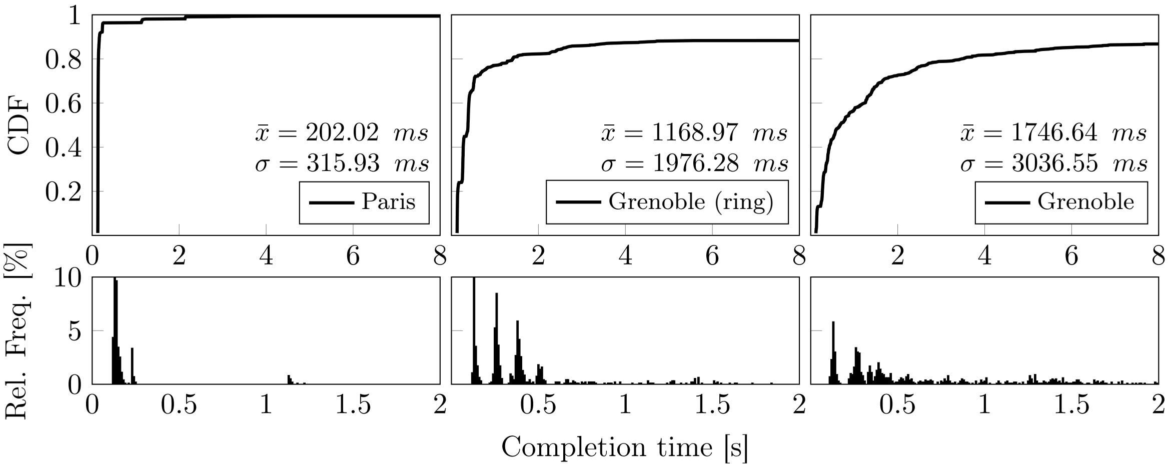


Figure 6.6: Time to issue alerts from publisher nodes to subscribers via the Content Proxy.

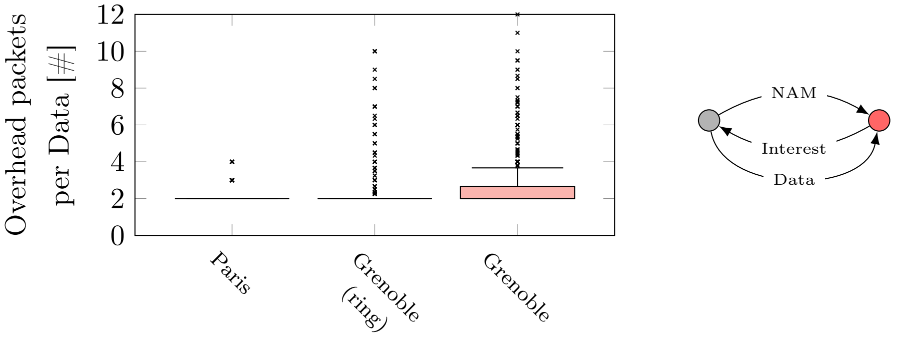


Figure 6.7: Overhead packets per publish event normalized by the hop count of a publisher for the three protocol deployments. The optimal overhead packet count is two (NAM and Interest).

hop WPANs. To a certain degree, we account this for the robustness of our hopwise content publishing and replication protocol. Further evaluations [111] confirm these observations.

Finally, we inspect the network overhead of publish operations for the three selected deployments. HoPP publishers replicate content hop-wise to parent nodes until they arrive at the content proxy. During this process, the actual data packets follow a sequence of NAM and Interest messages, *i.e.,* in the optimal case without any packet loss, the network overhead on a single link per successful content transfer consists of two messages. On packet loss, *e.g.,* due to saturated link resources or wireless interferences, the network overhead increases as the NAMs and Interests are retransmitted.

We count the number of distinct packets for all publish operations across all nodes in a specific topology, normalize them by the hop count for each node, and display the key statistical properties in Figure 6.7. While we observe a median of two for all three topologies, the statistical

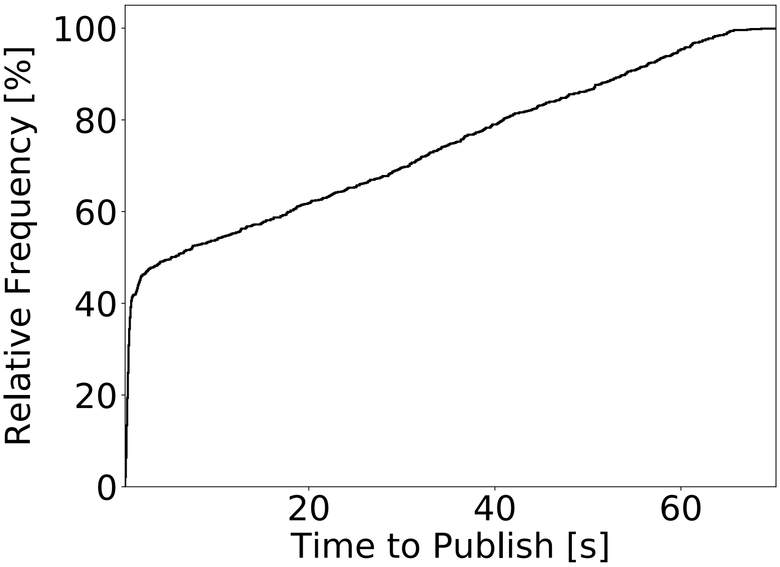


Figure 6.8: Time to content publishing at network partitioning.

dispersions show noticeable differences among the varying deployment options. As already observed in the former evaluations, the Paris topology experiences the least network stress and hence shows an overhead distribution centered around the median of two without any virtual variability. A few outliers indicate an increased overhead for two publish operations. The Grenoble (ring) topology increases in node density and path stretch, which marginally impacts the overhead: the variability around the median is still minimal, but more outliers indicate additional retransmission events. For the Grenoble deployment, we observe an enlarged spread where the middle 50% of measured data reaches an overhead of two to three packets. The outliers considerably increase, which signifies more network stress and is hence in line with the previous protocol evaluations.

6.4.4 Performance evaluation of mobility & network partitioning

We analyze a scenario of network partitioning on the Grenoble ring topology. To quantify the effects of a major network disruption, we disabled all nodes that are two hops away from the Content Proxy every 60 s for an off-time interval of 60 s. This isolated the Content Proxy periodically. Content publishing proceeded randomly with a frequency of one per 30±15 seconds.

Results in Figure 6.8 highlight a smooth content transition to the CP with a timing almost linearly stretched over the 60 s off-period. No unexpected content delays become visible, which indicates the protocol robustness on this macroscopic time scale.

The illustration in Figure 6.9 provides a more detailed overview on the number of publish events per hop and the time needed to replicate content objects to the immediate parent node. For an enhanced visualization, a single dot in Figure 6.9 represents ten publish events during

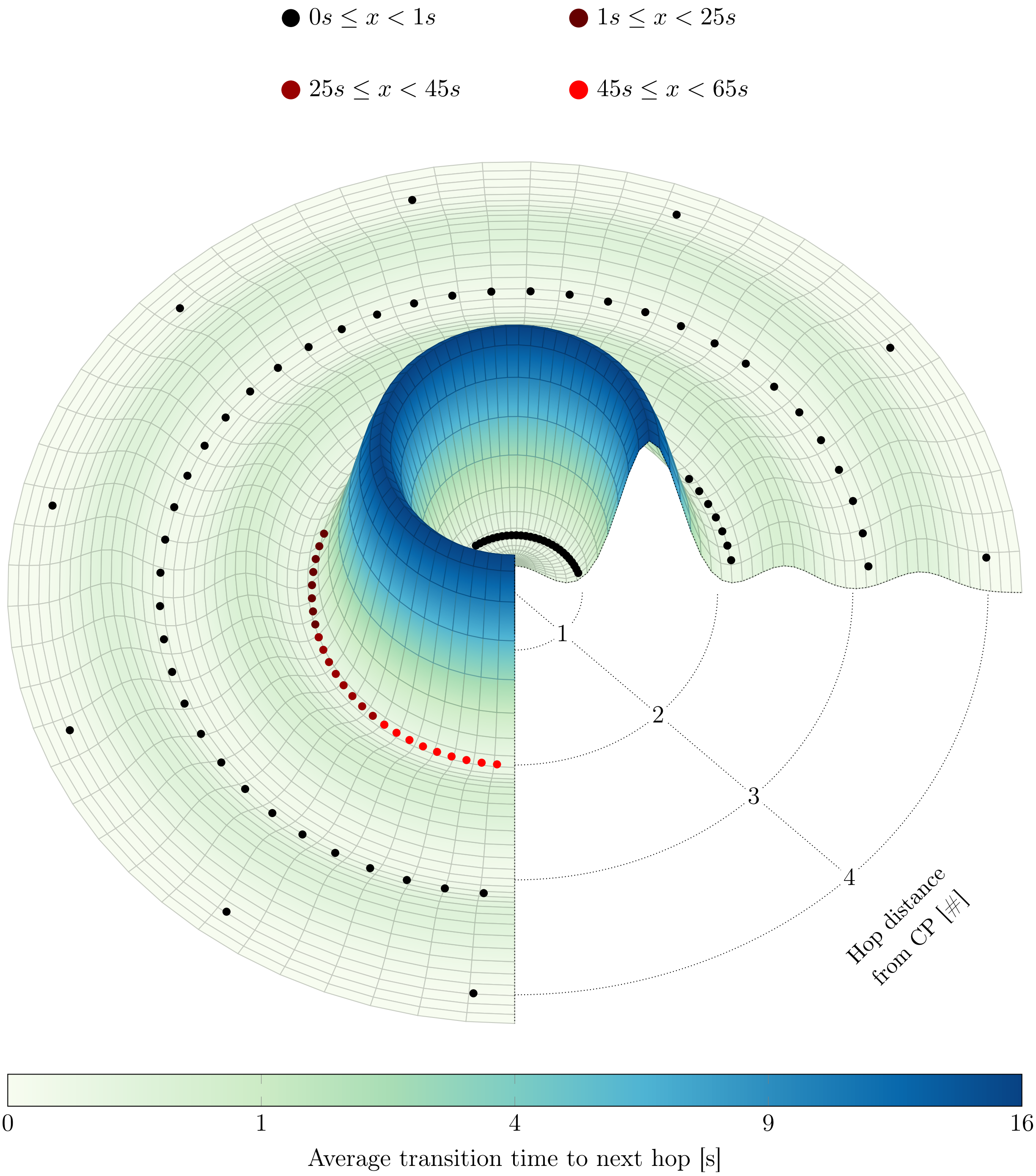


Figure 6.9: Visualization of publish events and publish time between hops in a partitioned network. Rings denote the hop distance of publisher nodes to the Content Proxy, which is situated in the center. Dots represent publish events originating from a node on a ring towards a parent node (next inner circle). Wave amplitudes and color codes of dots indicate the publish time to the next hop (black is quickly published, red has long buffer periods).

the duration of the experiment. Each dot is situated equidistantly on a ring, which symbolizes the distance of the publisher to the proxy node, *e.g.,* four hops for the outermost ring. Inner circles show more events due to the accumulative nature of publishing towards a single root node in the tree topology.

Waves between rings indicate the average duration of a content to be replicated to the next hop. The publish operation consists of the three messages NAM, Interest, Data, and needs 200–500 ms to finish for the majority of the events between rings 4→3, 3→2, and 1→root. Similar numbers were also recorded in our previous measurements (Figure 6.5). While waves between these rings show the same amplitude, an exception occurs between hop two and hop one: due to the configured partitioning, we observe much higher replication times, as already seen in Figure 6.8. ≈50% of all events between rings 2→1 show timings that increase from a few seconds up to 65 seconds. This gradual increase is depicted in the color coding of the dots. Darker events suggest a sub-second replication time, whereas red events indicate times up to a minute, in which case they are buffered for longer periods until the network reconnects.

6.5 Protocol Comparison

In this last part, we compare the memory requirements and incurred signaling overhead of HoPP with alternative publish-subscribe and mobility approaches.

6.5.1 Qualitative memory assessment

Varying publish-subscribe solutions show different memory requirements to store and maintain forwarding states on producer, consumer, and forwarder nodes. Since main memory is scarce on typical class 2 devices, schemes with high memory demands greatly impact the deployment scalability. In this comparison, we theoretically assess the state space for different publish-subscribe protocols (see Table 6.2).

First, we analyze necessary states to forward Interests (FIB) and Data (PIT) using the pure NDN protocol as baseline. The most decisive part affecting RAM usage is the name component: a hierarchical, descriptive naming scheme may include hash-based device identifiers, key fingerprints to handle access control, and timestamps to denote content recency. Requiring around 100 bytes per forwarding entry including face information like next-hop hardware addresses is therefore not unusual. For global producer reachability, each consumer maintains forwarding rules in the FIB commonly installed by an orthogonal routing protocol. While a highly hierarchical naming scheme may allow forwarding states to aggregate, space demands increase linearly with the number of published names ($\mathcal{O}(\mathcal{F})$) in the worst case. In contrast to consumers, producers do not require additional forwarding state due to the reverse path forwarding logic. Forwarders maintain reachability state for each name ($\mathcal{O}(\mathcal{F})$) and further maintain soft-state for each pending Interest ($\mathcal{O}(\mathcal{P})$) to preserve reverse paths. The linear increase in $\mathcal{F}$ and $\mathcal{P}$ is again most distinct in scenarios where state aggregation is not possible.

Protocol	Forwarding state requirements					
	Consumer		Producer		Forwarder	
	FIB	PIT	FIB	PIT	FIB	PIT
NDN	$\mathcal{O}(\mathcal{F})$	—	—	$\mathcal{O}(\mathcal{P})$	$\mathcal{O}(\mathcal{F})$	$\mathcal{O}(\mathcal{P})$
HoPP	$\mathcal{O}(1)$	—	$\mathcal{O}(1)$	—	$\mathcal{O}(1)$	$\mathcal{O}(\mathcal{P})$
PubSub-Mob [168]	$\mathcal{O}(\mathcal{F})$	—	—	$\mathcal{O}(\mathcal{P})$	$\mathcal{O}(\mathcal{F})$	$\mathcal{O}(\mathcal{P})$
NDN-Lite Pub-Sub [187]	$\mathcal{O}(1)$	—	—	$\mathcal{O}(1)$	—	—

Table 6.2: Theoretical space complexity analysis, where $\mathcal{F}$ denotes the number of forwarding entries (FIB) and $\mathcal{P}$ denotes the number of pending requests (PIT).

HoPP operates in lossy networks consisting of low-end IoT devices and decouples producers from consumers by using CPs as anchor nodes. As a consequence, HoPP requires only a single forwarding entry that points towards a CP on all devices with constant memory use ($\mathcal{O}(1)$). Additionally, forwarders maintain short-lived PIT state for subscriber Interests, which linearly increases with the number of pending requests ($\mathcal{O}(\mathcal{P})$). The number of available content publishings and subscribers does not affect the state requirements on producers and consumers, which is a necessary requirement for large-scale deployments.

Forwarding state in PubSub-Mob [168] is maintained and distributed via the external routing protocol NLSR [208]. Consumers and forwarders exhibit similar memory needs as a pure NDN deployment. Since this alternative publish-subscribe mechanism makes use of persistent PIT entries on producers, these nodes additionally store soft-state for the duration of each subscription event, *i.e.,* pending Interest.

NDN-Lite Pub/Sub [187] assumes all nodes to reside in broadcast range, so single forwarding entries can map directly onto the wireless broadcast address. This approach shows the lowest memory demands, but only works well in small-scale single-hop deployments. A broadcast communication typically lacks corrective actions on the link-layer, *i.e.,* timeouts and retransmissions, and is generally discouraged in low-power networks with numerous network participants in the same wireless domain due to uncontrolled interferences.

6.5.2 Signaling overhead and handover delay assessment

We now theoretically inspect the incurred signaling overhead that results from device mobility by comparing relevant protocol features of HoPP, MAP-Me [167], MIPv6 [207], and its anchor-based multicast adaptation M-HMIPv6 [209]. To analyze handover effects and quantities, we

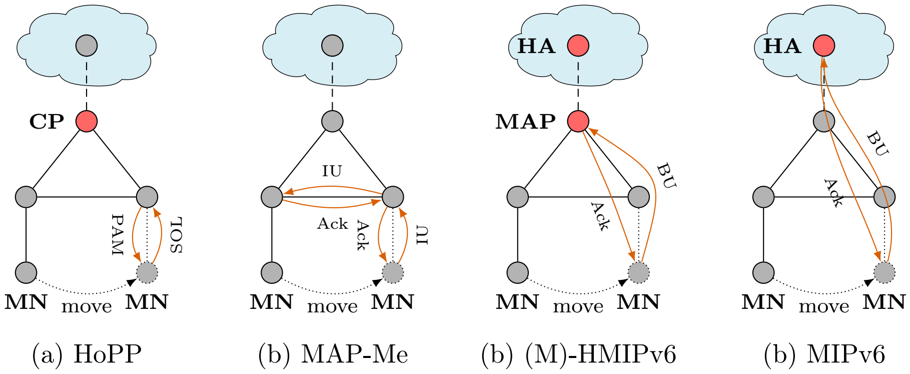


Figure 6.10: Topology setup and mobility-related signaling for the selected protocol ensemble.

assume a basic network topology where a single mobile node (MN) moves from a previous access router (PAR) to the next access router (NAR) as illustrated in Figure 6.10. In case of (M-)HMIPv6, we consider the path stretch from PAR to NAR as shorter than to a home agent (HA). This simple setup seeks to infer insights for the handover mechanisms rather than develop a complete, but complex quantitative picture. Figure 6.11 groups the signaling delays for the elemental protocol steps into three categories: (*i*) signaling that is geometry independent and occurs on a single hop, (*ii*) regional updates between the previous and current attachment points, and (*iii*) performing geometry dependent, global updates.

The selected protocols have different areas of application and therefore show varying implications on the link-layer. HoPP is designed for low-power multi-hop mesh networks without device associations on the link-layer. (M-)HMIPv6 and MAP-Me pursue general purpose deployments, which often follow star topologies when wirelessly connected. WiFi, as an example, requires device associations to an access point if not run in ad-hoc mode, which need to be re-established on device mobility. The layer 2 handoff time highly depends on the underlying link technology and hardware, but commonly approximates a delay of a few tens of milliseconds [209]. After successfully associating on the link-layer, an IPv6 node performs at minimum a local router discovery and an address configuration. The neighbor discovery protocol (NDP [210]) issues a link-local router solicitation (RS), which triggers a router advertisement (RA). Typically, these message exchanges yield a delay that is below 10 ms for most general link technologies with an exception for timeslotted solutions. HoPP manages its own prefix-specific routing system rooted at the Content Proxy (CP) and performs a similar task as soon as a node registers mobility: A scoped broadcast solicitation message (SOL) is transmitted to trigger a PAM from an upstream node. Similarly to RPL [202], a joining node inspects the ranks of multiple PAM responses to decide on the default router and then attaches to a particular DODAG branch. Here, delays similarly sum up to roughly 10–30 ms even with IEEE 802.15.4 as indicated in Section 6.4.3. MAP-Me does not specify the discovery process of default routers which suggests that it either

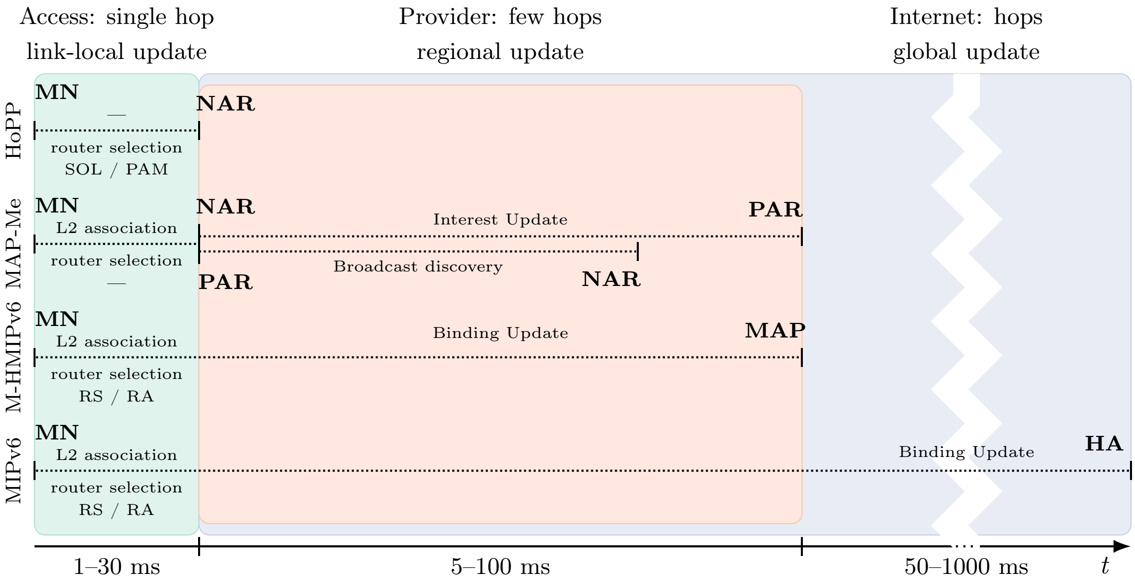


Figure 6.11: Semi-quantitative comparison of handover signaling delays for a mobile node (MN) on device mobility from a previous (PAR) to next (NAR) access router using HoPP, MAP-Me, (M-)HMIPv6, and MIPv6. Signaling group into local, regional, and global updates.

relies on an orthogonal routing protocol to install the correct next-hop, or that it configures the hardware address of the associated peer as a next-hop.

After completing device re-associations and local configurations, (M-)HMIPv6 notifies the Mobility Anchor Point (MAP) about node mobility by sending a *Binding Update (BU)*. Respectively, MAP-Me signals the new location to the previous access router by sending an *Interest Update (IU)* to the previous location. In our scenario description, we consider device mobility within regional scope, *i.e.,* the new access router is only a few hops from the previous attachment point away. A delay in the range of a couple of tens of milliseconds—more in case of intermittent link failures, especially on wireless media—is reasonable to assume. In contrast to (M-)HMIPv6 where BUs traverse end-to-end, the adaptive forwarding nature of NDN enables hop-wise state updates. While IUs propagate to the previous access router, any visited forwarder potentially installs the recent forwarding entry and can already divert ongoing traffic on path intersections to the new device location. Additionally, MAP-Me employs a quick discovery scheme to find breadcrumbs of disassociated mobile devices on neighboring attachment points by broadcasting ongoing traffic into the vicinity. If MNs leave breadcrumbs on attachment points in close range, then the consecutive broadcasting towards a specific MN can drastically decrease handover delays until a regional IU succeeds. On the other hand, defaulting to an unregulated broadcast is especially in dense wireless deployments harmful due to increased interference and missing retransmission features by the link-layer.

HoPP	MAP-Me	(M-)HMIPv6	MIPv6
10–20 ms	30–60 ms	30–60 ms	> 60 ms

Table 6.3: Handover latency ranges for mobility protocols based on the topological assessment in Figure 6.10.

MIPv6 does not employ similar shortcut methods to decrease the handover time, but rather relies on the geometry dependent binding update to the home agent. Since the path stretch between a new device location and the home agent may consist of an indefinite number of hops, the update procedure may require a period ranging from a few tens of milliseconds up to the order of seconds, in which a mobile node is unreachable. Conversely, HoPP does not require global routing updates for mobile devices since producers publish content per prefix to a content proxy that anchors the prefix in the underlying routing system. Subscribes retrieve data from content proxies, which decouples them from the actual producers. Data objects are hence still accessible even in the event of longer device sleep cycles or intermittent connectivity issues due to mobility.

In the following analytical evaluation of handover latencies, we assume the simple topology depicted in Figure 6.10. For comparability reasons, the nodes in all protocol deployments connect in an ad-hoc fashion via IEEE 802.15.4, so we do not apply any association times on the link-layer when moving to a new access router. Table 6.3 summarizes the handover latency range for a single mobile node (MN) that moves from a previous access router to a new location. On a successful move, the MN performs a local router discovery and potential network configurations. Typically, this process requires at least one message round-trip (SOL-PAM, RS-RA) initiated by the MN. Former experimental testbed evaluations [111] indicate a single-hop round-trip time of roughly $10 - 20$ ms with similar radio configurations. In addition, MAP-Me and (M-)HMIPv6 perform a regional update that reaches two hops in our analytical model. Based on the previous round-trip assumptions, this adds another $20 - 40$ ms to the total handover latency. In the case of MIPv6, the Binding Updates are geometry dependent and traverse an indefinite number of hops, which can add a significant delay to the handover. Additionally, security considerations for (H)MIPv6 updates may require protective measures, *e.g.,* with IPsec, especially if they have global reach. While an increased packet overhead due to security features is insignificant on the Internet site, it is much more impactful on the IoT domain. Packets surpassing the maximum transmission unit (MTU) for IEEE 802.15.4 of 127 bytes get hop-wise fragmented by the 6LoWPAN [3] convergence layer and add further round-trips per hop for each fragment.

6.6 Conclusions

Node mobility and intermittent connectivity in low-power regimes severely challenge the routing between sensors and actuators. Long handover delays can result in extended downtime of nodes, or even partition a network topology. In this work, we found that (a) publish-subscribe with named topic prefixes can overcome the complexity of routing named data of things, and (b) NDN with *link-local* alerting has striking advantages for reactivity, security, and robustness in constrained environments.

We introduced the lightweight publish-subscribe system HoPP that was implemented in the CCN-lite network stack adaption of RIOT and experimentally evaluated in large, realistic testbeds with varying topologies. Our findings confirmed that the HoPP approach is robust and resilient while performing well in the majority of experiments. In particular, we could show that node mobility and temporary network partitioning require low repair overhead and can be quickly mitigated by local buffering and re-connects.

Part II

A Data-centric Web of Things

Chapter 7

Motivation and Problem Statement

7.1 Lessons Learned From a Decade of ICN Research

ICN research primarily applied information-centric concepts—(*i*) name-based, stateful forwarding, (*ii*) in-network caching, and (*iii*) content object security—to wired network devices with significant memory and processing capacities. The intention was to ease content retrieval in view of growing demands on the Internet, and shield the network infrastructure against distributed DoS attacks. A subset of these ICN concepts were formerly used in energy preserving approaches to reliably *diffuse* [211] sensor values in Wireless Sensor Networks (WSNs), before their wider application gained popularity with common ICN technologies. Work on ICN IoT deployments [190, 110] began to explore these ICN principles in regimes of low reliability, and the discussions in Part I of this manuscript reveal several lessons for operating CCNx and NDN in challenged networks of wirelessly connected *things*. In the following, we summarize these lessons and highlight our observations.

Stateful Content Replication. Packet transmissions are susceptible to loss due to radio interference and saturated device resources, especially for congested multi-hop configurations in which error probabilities accumulate with every hop. Corrective actions are required to recover from packet loss in challenged regimes, but these additional packets further add link stress and yet lead to even more packet loss. The replication of content objects in ICN deployments relieves network stress by shortening request paths due to caching and by reducing the overall number of packets due to request aggregation. As stateful forwarding and hop-wise caching further loosen the coupling between content and data producers, networked *things* can benefit from asynchronous access to content. They hand over the responsibility for content objects to the network and can act autonomously according to local sleep cycles to reduce battery use.

In contrast to the stateless best-effort forwarding of IP-style communication, this replicative delivery of data can, however, put pressure on memory demands. Forwarding and caching is typically situated on the main memory, instead of secondary storage. The advantage of SRAM over nonvolatile storage, *e.g.,* flash memory and SD-cards, is its fast I/O access, low energy demand, and long lifespan, but it is also more expensive. To keep unit prices low, main memory is therefore severely limited in size to 32–128 KiB for cheap *things*, which is shared between OS, network stack, and application needs, while secondary storage can range from megabytes to gigabytes. A few request paths and data cache entries can already saturate this memory space, and a lightweight management of resources that balances between main memory and secondary storage to lessen power consumption is therefore obligatory to ensure efficient network functionality.

Content Naming. Named-Data Objects (NDOs) are the centerpiece of information-centric networking. The focus on names from network layer to applications removes the need for additional infrastructure, such as the Domain Name System (DNS). Names are of variable-length and can include application-specific details, while they allow for self-identifying content objects and enable the in-network processing on the forwarding fabric. One prominent example for application details in names appears in typical sensing use cases, where devices periodically generate readings and publish new content objects. To denote sensing recency and to ensure the uniqueness of names, devices can include sequence numbers that increment with each sensing operation as part of the naming scheme. However, this approach requires the persistence of counters, since uncontrolled system reboots reset counter state and then lead to increased chances of name clashes with previously generated NDOs. Another approach without handling counter state is to utilize global time in names, *e.g.,* the time since Unix epoch, instead of locally incrementing counters. While the latter approach does not require an energy demanding persistence of state, it needs either additional network complexity to synchronize network time (*e.g.,* NTP), or an appropriate hardware module that provides a time reference, such as a GPS receiver. Lengthy names can impact the network performance as requests and responses contain them for forwarding purposes. Thus, a naming scheme for IoT use needs to carefully consider a design that optimizes for small packet sizes and for a small memory consumption in the forwarding-related data structures: Content Store, Forwarding Information Base, Pending Interest Table.

Pull-based Communication. The request-response paradigm of prominent ICN flavors, their in-network caching, and the absence of node addresses are fundamental properties to increase resistance against distributed DoS attacks. This especially holds for the IoT as the number of connected *things* is steadily growing. A pull-based communication by design prevents the dissemination of unwanted data, and while it is fitting for retrieving existing or scheduled content, it requires more complexity with unscheduled and absent data. A naïve polling of yet unpublished content is the simplest method to discover and retrieve unscheduled data, but it also introduces network stress. Especially in CCNx and NDN, where a reverse path is constructed for returning responses, many distinct requests can quickly exhaust limited device resources, which may lead to incoherent forwarding states [127, 39]. Alternative solutions in research enable an unsolicited delivery of data with (*i*) long-lived request states that persist for multiple returning responses, and (*ii*) custom push primitives that distribute the data over multiple hops to a designated destination. While both approaches may require no or only a few additional round-trips and thus effectively reduce the number of message transfers, they also introduce a range of problems. Receiver mobility, *e.g.,* requires a more complicated routing, whereas it is inherently supported by the pull-driven communication of ICN. Sender-initiated data traffic can further degrade the network performance by poisoning on-path caches with unwanted NDO copies. Unsolicited propagations generally need fewer message transfers compared to retrievals, but this does not apply in regimes of low reliability. It is common practice to exchange positive acknowledgments also for unsolicited packet transfers, like for confirmable CoAP PUT messages or reliable MQTT-SN publications. Compared to these schemes, the round-trip of a request-response transaction amounts to the same number of hop traversals.

Content Object Security. The stateful forwarding and in-network caching foster a security model that applies to the content itself, instead of the transport session. Security on the content object level brings advantages to the low-power domain: Contrary to secured transport sessions where expensive cryptographic operations are performed synchronously for each packet, the resource-constrained *things* only need to protect the data *once* with content object security. Data is asynchronously protected prior to the actual retrieval, and devices can schedule these operations when power is available, *e.g.,* in energy harvesting scenarios with volatile and periodic energy supplies. Despite its benefits, the computational effort scales with the frequency of content generation. To reduce energy cost, symmetric cryptography is favored over asymmetric cryptography, which is considered too expensive for configurations without hardware assistance. A viable alternative to digital signatures with asymmetric crypto is the Keyed-hash Message Authentication Code (HMAC). This choice also mitigates the need for a public key infrastructure, but requires the distribution of pre-shared secrets via out-of-band channels. Elaborate schemes may further reduce the computational effort by batch processing data aggregates, or by using sophisticated data structures like Merkle trees to sign and verify continuous data blocks as illustrated by NDN DeLorean [212].

7.2 Deployment Barriers in Heterogeneous Protocol Landscapes

In Part I, we focused on protocol operations within IoT stub networks to gain insight in the strength and weakness of host-centric and information-centric protocols. Next, we want to highlight deployment effects and concerns that surface when these IoT regimes connect to a cloud infrastructure on the Internet.

The common IoT deployment as illustrated in Figure 7.1 consists of many smart *things* connecting via a multi-hop configuration (*e.g.,* smart factory) or a star topology (*e.g.,* smart home) to an application layer gateway that transforms protocols and payloads. These gateways further connect to vendor cloud services, which then perform mission-specific data manipulations for customers. Cloud interactions are mainly based on a RESTful access via HTTP, or on a publish-subscribe system with MQTT. Both mechanisms use TCP for reliable data exchanges between IoT edge gateways and cloud services on the Internet. For the low-power regime, TCP is typically considered too complex, which is why 6LoWPAN-based setups prefer the much

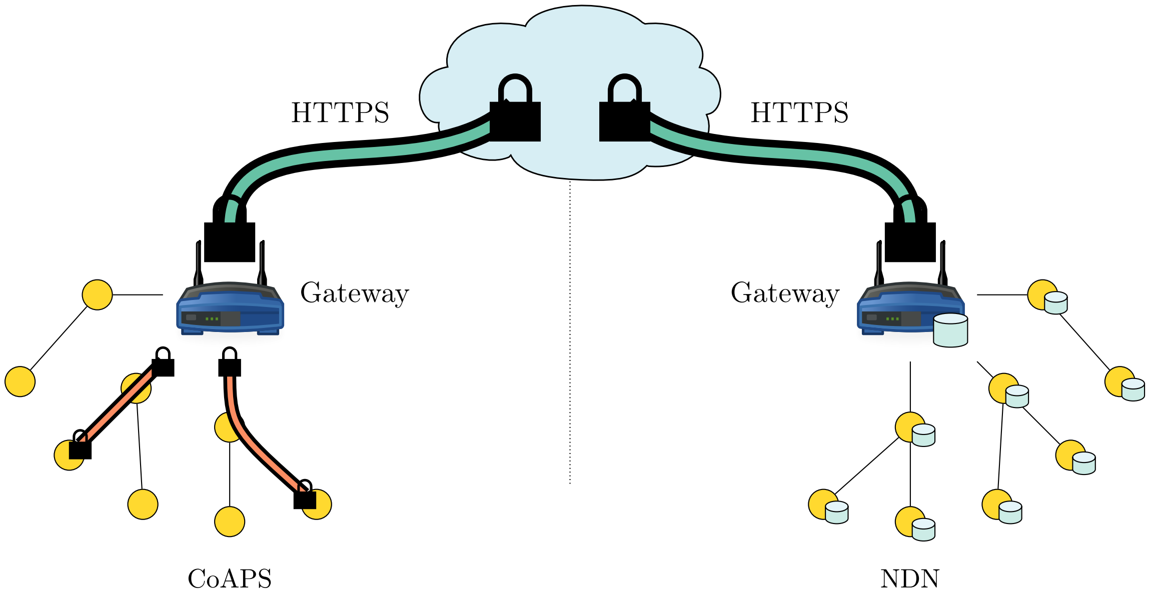


Figure 7.1: The complete deployment view for the host-centric and information-centric deployment options.

simpler UDP transport, and the RESTful application layer protocol CoAP substitutes HTTP to benefit from its space efficient protocol encoding. An ICN deployment can use an overlay solution to drive the IoT, but this (*i*) increases memory demands on devices for operating a heavy dual stack solution, and (*ii*) inflates packets due to the protocol nesting. Replacing the network layer with CCNx or NDN in conjunction with the ICNLoWPAN link convergence layer yields a more efficient deployment option for the IoT domain, but also requires more complex logic on gateways. In the following, the selected deployment options are qualitatively compared under various aspects to assess their operational capabilities.

Gateway Complexity and State Demands

Host-centric Deployment. The Internet design follows an end-to-end principle [63], where transport sessions establish directly between applications without intermediaries. However, transport conversions, *e.g.,* between UDP and TCP, require transitional application layer gateways that break this end-to-end transport by terminating application traffic between IoT devices and cloud services. These middleboxes store endpoint information for active protocol transactions, *e.g.,* for CoAP requests that await returning HTTP responses from cloud applications. While uplink state on gateways is normally limited to a few preconfigured cloud endpoints, the IoT-facing side requires endpoint state that scales with the number of *things* and ongoing traffic flows. CoAP and HTTP both implement the REST paradigm and are intentionally very similar in their design. Hence, the proxying functionality between these application layer protocols is straightforward, requires less computational effort, and enjoys a wide adoption in network software libraries.

Information-centric Deployment. Native ICN deployments demand for complex protocol conversions, since not all communication methods are easily transferable. RESTful HTTP requests can easily be mapped to ICN requests, but *alerting*, *e.g.,* is not as trivial as with an IP-style communication. HTTP POST and MQTT publications require an elaborate transport assistance on gateway nodes, particularly due to the missing concept of source node addresses on the ICN side. The ICN routing system needs adjustments to support the addressing of designated endpoints. This is achieved by either installing appropriate endpoint-based forwarding states, or by maintaining and refreshing receiver-initiated reverse paths. In addition, there is no standardized support for a proxying functionality to HTTP in widely adopted network software libraries that integrate into the CCNx or NDN stack, which impacts the acceptance for such heterogeneous deployments. However, while the computational effort for protocol conversions may be larger than in IP-based deployments, state demands on gateways can also be smaller in multiparty scenarios. The name-based and stateful forwarding natively supports the aggregation of requests and the fan-out of responses. This limits state to the number of unique content requests independent of the number of actual *things*.

End-to-end Security

Host-centric Deployment. The break of transport sessions is necessary to enable connectivity in situations of heterogeneous protocol landscapes, but it also affects end-to-end security. Transport security is the prevalent method on the Internet to protect transport sessions between individual endpoints. For a continued protection, security mechanisms need to range from cloud services to applications running on the IoT devices, but application layer gateways harm this end-to-end principle by interrupting the transport. These middleboxes need to be included in trust relationships to re-encrypt and re-authenticate packets, and thus guarantee the confidentiality and integrity of content between IoT devices and vendor applications. As an additional entity of the infrastructure, however, security related risks increase: gateways may be compromised, or they may unintentionally leak data in an unauthorized manner.

Information-centric Deployment. CCNx and NDN deployments use security on the content object level. This ensures an end-to-end protection beyond potentially untrusted application gateways. However, one drawback of the security envelope for requests and responses is that they cannot be transformed on middleboxes without compromising the end-to-end trust by having a *man-in-the-middle*. An easy solution is to nest ICN packets in an IP-based overlay network, but this increases bandwidth demands and exposes cloud applications to the ICN technology. The latter complicates mission-critical application code on the vendor and customer side. Content object security therefore seems more practical in homogeneous protocol deployments, which require no further protocol or payload conversions.

Interoperability with Internet Services

Host-centric Deployment. Convergence via the network layer and a layered composition of protocol operations are major advantages of the Internet architecture. 6LoWPAN setups are theoretically compatible with the full range of protocols that run on IPv6, although in practice many are challenged by the resource-constrained nature of low-power networks. This universal integration level increases deployment agility and allows for the easy application of utility protocols, such as ICMPv6 for error reporting and debugging, DNS for resolving domain names, or NTP for synchronizing with Internet time. For IoT deployments that natively support the correct transport, no application layer gateways with complex protocol conversions are required. Border routers are still necessary for link convergence, but these operations are transparent to the network, transport, and above.

Information-centric Deployment. The network layer substitute of CCNx and NDN IoT deployments complicates the use of prevalent utility protocols and Internet services. Two issues exist that application layer gateways can try to address. First, transformations on the packet level are mandatory for protocol intelligibility. Second, specific protocol mappings need to be defined, and the protocol literacy of devices on the IoT side has to be updated for the adapted protocol logic. This part is cumbersome and impedes the integration of alternative protocols on evolving requirements. At the same time, conventional infrastructure elements like corporate firewalls, load balancers, and components for handling traffic monitoring, prioritization, bandwidth throttling, and rate limiting of unauthorized transactions are inaccessible without drastic adjustments to the application logic and the transport.

Deployment Résumé

The need for transport transformations puts deployment barriers for IP-based as well as ICN-based network configurations in place. Gateways are infrastructure entities that can address these barriers with application-specific protocol conversions, but these middleboxes interrupt the host-centric end-to-end transport and restrict security guarantees. Trust relationships are bend to include (potentially third-party) gateway devices. Security on content object level as employed by the ICN flavors proves valuable in situations where trust cannot be established with gateways, but also obligates an overlay solution and defers any necessary protocol translation to the actual cloud endpoint. While Part I has shown the promising benefits of information-centric principles for low-power regimes, the complete deployment picture reveals challenges of deployment agility and a burdensome integration effort to ensure interoperability with the Internet. This calls for a deployment option that (*i*) brings information-centric characteristics into the IoT, and (*ii*) enables a seamless integration with Internet services.

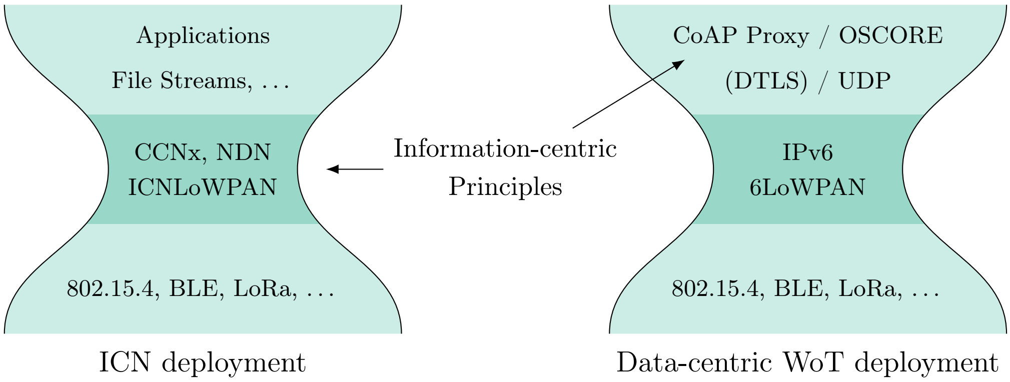


Figure 7.2: For ICN deployments, information-centric principles are located on the network layer, while they reside on the application layer for the data-centric WoT deployment.

7.3 A Data-centric Deployment with Internet Perspective

Native ICN deployments use information-centric principles on the narrow waist of the network stack (see Figure 7.2), and require less infrastructural complexity than IP counterpart deployments. This allows for lean software components that seamlessly fit into the limited capacities of resource-constrained devices. However, one downside of this integration level is its protocol incompatibility with Internet services, which causes a fragmented deployment landscape. Complex transformations on gateway devices can enable interoperability at the cost of deployment agility. This dissatisfying state gets worse when deployments use third-party gateways, which vendors cannot manage on-demand and do not include in their trust relationship. Challenged IoT configurations call for alternative paths that enable the efficacy and reliability of the information-centric paradigm, while retaining interoperability with Internet services.

With an intensified umbrella effort on adapting existing, standardized web technologies for IoT use, the standards organizations World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) provide the protocol fabric for an efficient and interoperable Web of Things (WoT). The IETF CoRE working group has recently developed a rich set of additional features for CoAP, which open various deployment options—stateful forwarding, in-network caching, and content object security are among them. These emerging building blocks of the CoAP protocol suite as depicted in Figure 7.2 can harmonize the deployment landscape and provide an Internet perspective by building a RESTful WoT that adheres to ICN first-hand principles and its performance on the application layer. This approach enlivens the hope to take advantage of the various insights and techniques that emerged from ICN research and leads them into a promising, realistic deployment trail for the fast emerging IoT.

In Chapter 8, we showcase a firmware update scenario to motivate the need for a data-centric WoT. Chapter 9 evaluates the two security models (*i*) transport layer security and (*ii*) content object security for the IoT. In Chapter 10 and Chapter 11, we discursively summarize the problem space of (multiparty) content retrievals in low-power and lossy wireless networks, and identify a suitable feature set of standard CoAP protocol elements to build a WoT that is data-centric, but still compatible with classic CoAP deployments. The overarching road-map of Part II includes the construction of a CoAP deployment option with a name-based and stateful forwarding, hop-wise caching, security at the content object level, and inherent support for group communication.

Stateful Forwarding and Naming. At the center of this data-centric deployment construction is the CoAP proxy functionality. Much like general proxy servers, a CoAP proxy performs application logic as an intermediary between a client requesting a resource and a server. The typical use case envisions a CoAP proxy at the Internet edge, most likely on a gateway node that converts between CoAP and HTTP. By placing a proxy node on multiple hops along a path in the IoT domain, however, a deployment option emerges that closely complies to the design considerations of CCNx and NDN. This not only makes each forwarder application aware, but also introduces a name-based forwarding that liberates content and service URIs from host addresses.

Hop-wise Caching. The inherent support for response caches on proxy nodes takes the deployment construction further towards a data-centric WoT as it (*i*) contributes to the loose coupling of content objects and their origins, and (*ii*) increases the reliability of content retrievals by shortening request paths. One important point that requires special consideration concerns the validity of cached content objects. While NDN imposes immutable bindings of names to content objects to achieve a long cache liveliness, easy validation of content provenance, and protection against delayed and replayed messages, classic CoAP requests may return responses with dynamic payload. To benefit from the same advantages, CoAP applications can encode the variability of content into resource URIs, such that each content object is bijectively mapped to exactly one URI.

Content Object Security. The next step on the data-centric road-map is the introduction of a security model that is in agreement with the approach to a cachable content access. Content object security as provided by OSCORE fills this gap. It maintains an end-to-end protection of content objects beyond untrusted gateways by nesting the original CoAP message as an encrypted and authenticated OSCORE option into another CoAP message. As part of preventing replay attacks, the transactional request-response binding of CoAP is further bolstered in OSCORE. At the cost of these protective measures, the utilization of caches are burdened as requests cannot be served by matching responses in caches beyond transactional request-response bindings. Request retransmissions still benefit from shortened request paths due to cache hits on on-path caches, but subsequent OSCORE protected requests to the same resource—even for the same origin—generate unique cache keys, and thus cache misses.

Secured Group Access to Content. Multiparty content dissemination is an important use case in common actuator scenarios (*e.g.,* switching light bulbs) and in particular for over-the-air software updates. IP multicast, however, is difficult to implement in low-power, wireless regimes due to problematic layer 2 multicast mechanisms. In addition, the synchronous nature of IP multicast endangers successful packet transmissions due to interferences. Corrective actions to recover packet loss in multicast configurations usually (*i*) involve a substantial signaling overhead, or (*ii*) lead to redundant message deliveries. The NDN architecture, on the other hand, inherently supports the seamless replication of group content. By attaining a stateful forwarding and hop-wise caching, the data-centric WoT construction can replicate the same multiparty communication capability. However, similar to the previous problem of cache utilization, the strong message binding of OSCORE also burdens the protected group communication due to the transactional state that exists between two security contexts on two different endpoints. The current solution to this concern is to establish a security context that is shared among multiple endpoints in a group, and to relax the source authentication property of OSCORE, so that repeated requests to the same resource from multiple endpoints result in the same cache key. This admittedly weakens the protection against request replays on a CoAP server, but an operational group access to cached content on on-path caches already desensitizes applications to this type of attack.

Chapter 8

Use Case of Reliable Firmware Updates

Abstract

Security in the Internet of Things (IoT) requires ways to regularly update firmware in the field. These demands ever increase with new, agile concepts such as security as code and should be considered a regular operation. Hosting massive firmware roll-outs present a crucial challenge for the constrained wireless environment. In this chapter, we explore how information-centric networking can ease reliable firmware updates in edge networks as visualized in Figure 8.1. We start from the recent standards developed by the IETF SUIT working group and contribute a system that allows for a timely discovery of new firmware versions by using cryptographically protected manifest files. Our design enables a cascading firmware roll-out from a gateway towards leaf nodes in a low-power multi-hop network. While a chunking mechanism prepares firmware images for typically low-sized maximum transmission units (MTUs), an early Denial-of-Service (DoS) detection prevents the distribution of tampered or malformed chunks. In experimental evaluations on a real-world IoT testbed, we demonstrate feasible strategies with adaptive bandwidth consumption and a high resilience to connectivity loss when replicating firmware images into the IoT edge.

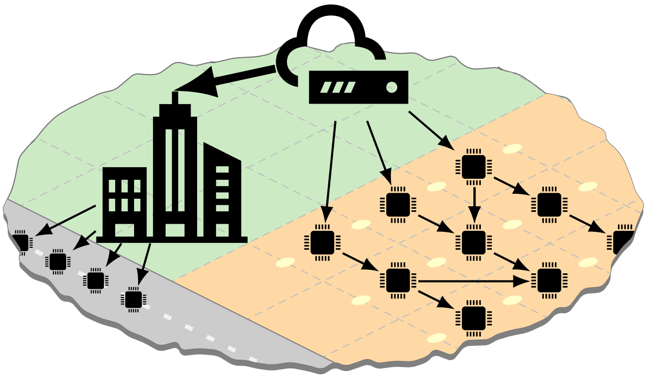


Figure 8.1: Massive firmware roll-out campaign in distributed and heterogeneous networks

8.1 The Problem of Firmware Propagation and Related Work

8.1.1 Challenges in low-power regimes

Secure firmware roll-out campaigns for large-scale IoT deployments demand a coordinated interaction with great regularity between multiple stakeholders. Vendors prepare and publish firmware versions and local site administrators oversee the roll-out procedure. An autonomous firmware update without physical proximity can drastically reduce the roll-out time and management overhead for local site administrators. IoT devices connect through low-power and lossy networks (LLNs) to powerful border routers. Especially in industrial and rural settings where infrastructure is challenged by natural and regulatory constraints, wireless multi-hop networks are prominent and continuous access to deployed hardware is not always feasible. These regimes are subject to radio interference and individual link error probabilities that accumulate in a destructive manner. In addition, limited maximum transmission units as well as low bandwidth and high delay link capabilities further complicate the distribution of large firmware objects, which necessarily split into hundreds to thousands of fragments. While corrective actions on the link, network, and application layer usually recover packet loss, small amounts of retransmissions behave additive and induce link stress in broadcast range, which impacts energy expenditures of battery-operated devices. The exhaustive task of delivering image files also opens up significant attack vectors for denial of service (DoS) attempts. Willfully tampered or inadvertently modified firmware images deplete network and memory resources to a point where devices neglect mission-critical duties.

The importance of well-thought-out firmware roll-out architectures that efficiently operate in low-power regimes and display a resilient security posture is undisputed. Several approaches have been proposed in research or have already been deployed in industrial solutions.

8.1.2 Firmware updates in the IoT

SUIT [213] is a recent addition to the menagerie of firmware update architectures. It is driven by the eponymous IETF working group and aims for a standardized update mechanism in constrained IoT networks that is reliable and secure. SUIT specifies a concise and machine-processable manifest document [214, 215] that describes meta-data of firmware images, such as their download location, firmware version, and optional processing steps to decompress and decrypt binaries. This architecture relies on the Internet protocol stack for retrieving updates and therefore expects certain protocol mechanisms to be present, like congestion and flow control, packet fragmentation, and the ability to resume corrupted transfers. Given its current momentum at the IETF, we consider SUIT as a suitable blueprint for our information-centric firmware update approach.

ZigBee [216] is a protocol specification harboring various network solutions to interconnect a wide range of heterogeneous, ZigBee certified devices. It builds on IEEE 802.15.4 and is prominently used by several product lines, such as Philips Hue, OSRAM lightify, and some Xiaomi devices, albeit not always securely [217]. In the ZigBee Over the Air (OTA) Upgrade Cluster module, clients regularly poll firmware information, or a server performs *Image Notify* push operations for clients not in hibernation. The distribution of upgrade images via broadcast or multicast is not recommended due to a missing point-to-point security. In this case, ZigBee advises a separate unicast attestation with the upgrade server after completing an image transfer. In contrast to ZigBee, we believe the vendor-independent manifest files of SUIT to concisely organize meta-data provide a greater accessibility to the update process in heterogeneous network deployments.

8.1.3 Reliable content transfers and data management in constrained networks

Large data objects, such as uncompressed binary images with moderate software complexity for embedded devices, can reach file sizes in the range of tens to hundreds of kilobytes. Prior to the IoT era, wireless sensor networks (WSNs) explored network reprogrammability of low-power devices in broadcast media and reliably disseminated large data objects (*e.g.,* a firmware) using epidemic routing methodologies [218, 219, 220]. Delicate adjustments to the classic flooding, such as node density awareness, windowing, the use of negative acknowledgments (NACKs), and unicast requests with broadcast data transmissions have shown promising results in lossy networks. Due to the generally inconsistent protocol layering in former WSNs, packets exceeding link MTUs in constrained network environments had to be fragmented on the application level.

In contrast, the current IoT mostly builds on IPv6 and to bypass transmission limits of typical link layers, the IETF designed 6LoWPAN [3]—a convergence protocol to adapt IPv6 functionalities to challenging LLNs. It supports a header compression to reduce header verbosity and a fragmentation scheme [3], which caps at 2048 bytes and is therefore inoperable for firmware propagations. The constrained application protocol (CoAP) [4] is part of the IETF envisioned IoT network stack and supplements IoT networks with a RESTful communication paradigm. Block-wise transfer [221] is an add-on to CoAP for splitting a payload into equally sized blocks, which are then iteratively transmitted with minimal server-side state. Chunking on the CoAP level further enables the use of CoAP reliability features for each separate block.

Recent studies [222, 111, 98] reveal a superior data delivery performance for named-data networking (NDN) [21] in low-power networks compared to end-to-end IoT protocols, such as CoAP and the message queuing telemetry transport for sensor networks (MQTT-SN) [6]. NDN leaves the fragmentation of larger named-data objects to upper layers, since naming decisions for newly created chunks are highly application-specific. Link fragmentation extensions [115, 223, 31] operate below NDN and modify the packet structure. Other approaches [224, 102, 225] apply a fragmentation and naming scheme on the application to yield a structured access to data chunks with predictable names.

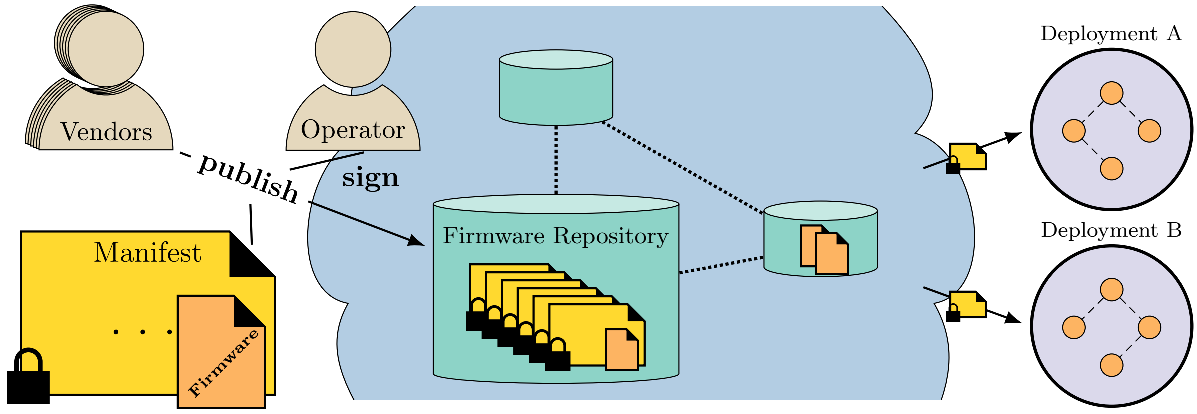


Figure 8.2: Overview on the back-end system of our information-centric, reliable firmware roll-out approach.

$$/\ \underbrace{\texttt{OilRig-3}}_{\text{Deployment}}\ /\ \underbrace{\texttt{IoTCompany-5}}_{\text{Vendor}}\ /\ \underbrace{\texttt{Valve-7}}_{\text{Device Class}}\ /\ \underbrace{\texttt{1632261600}}_{\text{Timestamp}}$$

Figure 8.3: Namespace schema.

8.2 Building Blocks for Reliably Updating Firmware with NDN

8.2.1 Roll-out campaign management

We design a secure and reliable campaign management system for firmware roll-outs that handles the delivery of software updates to numerous constrained edge devices in multiple sites using NDN. We use the SUIT [213] model as a blueprint for our information-centric approach and adopt essential system components and the same terminology. Figure 8.2 illustrates our name-based back-end proposal, which consists of three components: (*i*) publishing and versioning firmware images and manifest files by vendors, (*ii*) managing the storage of chunked software updates by an operator and providing access to the IoT deployment sites, and (*iii*) a timely notification of version updates and a reliable delivery of necessary updates towards edge devices on the IoT side.

8.2.2 Firmware preparation and publication

Namespace management. Large site deployments can consist of heterogeneous devices from varying vendors and the highest level of interoperability is essential to construct an energy-efficient system. A systematic namespace management regulates all interactions between vendors and IoT devices. Figure 8.3 demonstrates our name schema used for all components, ranging from upper-layer application functions down to forwarding and caching duties.

Each deployment has a globally unique name and may identify an offshore drilling rig, segments of a connected urban network, or a smart home environment. We consider the deployment identifier as the leading component in our name schema to keep forwarding states towards single deployment sites minimal, *i.e.,* they most certainly aggregate due to the spatial proximity of devices within the scope of a deployment. Vendor names are equally globally unique like deployment identifiers and both components are managed by the same, external registry. Finally, a device class designates a specific firmware instance for all nodes of the same product type. The timestamp component describes the actuality of a firmware and is encoded as a Unix timestamp with a predefined granularity. To fully leverage the in-network caching abilities of NDN, binaries are prepared for device classes instead of yielding unique binaries for each single device. This also reduces the binary management overhead on the vendor site.

Firmware generation. Vendors precompile firmware images for their deployed product lines and keep track of the software versioning. Since binaries are prepared for device classes, the images cannot ship with sensitive data. Device-specific configurations are rather obtained on run-time after a successful firmware installation on an IoT node. This requires that each IoT device is provisioned with vendor-specific data for bootstrapping purposes during the manufacturing stage or with the use of an out-of-band channel, which is already common practice for real-word deployments. This data outlasts firmware upgrades and is stored independently of the program code, *e.g.,* in a dedicated address space on the flash memory, or using an SD card. To protect the firmware integrity, vendors also generate a message digest of the binary alongside the firmware image.

Preparation of firmware chunks. The small-sized MTUs in common network link technologies disallows the transmission of images in single network packets. For a successful delivery, an image fragmentation at the vendor and a reassembly at the IoT edge devices is necessary. Fragmentation on convergence layers [3, 31] is a solution to provide a hop-wise, fragmented delivery between two peers, but due to the layering, these schemes make the caching of individual fragments impossible. Thus, we focus on a fragmentation approach that chunks the image on application level and reassembles them at the IoT edge device after all chunks have been successfully retrieved.

The reassembly of fragmented images must be as simplistic as possible for the constrained devices. We therefore follow a linear chunking of the image file in our solution, where each chunk is of fixed length (the last chunk being an exception). The reconstruction on the low-power devices is straightforward as fixed-length chunks can be joined using offsets, which makes the need for an ordered delivery unnecessary. Chunk sizes may vary between device classes, since different link-layers will yield different MTUs. Each chunk is addressed by appending a monotonically increasing chunk identifier */chunk/id* to the base name (see Figure 8.3), starting at */chunk/0.*

Manifest description. As demonstrated in Figure 8.4, a vendor also creates a manifest file following the SUIT model to organize meta-data on the firmware version and binary image.

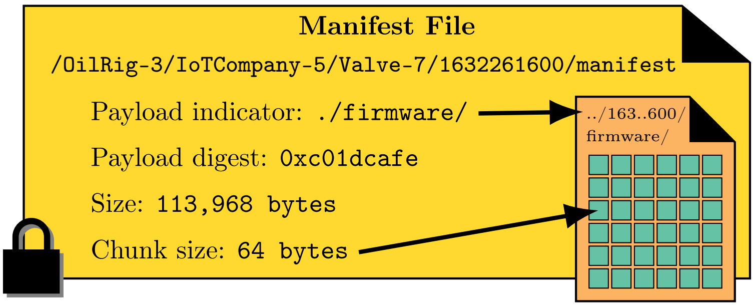

Manifest File
/OilRig-3/IoTCompany-5/Valve-7/1632261600/manifest
Payload indicator: ./firmware/
Payload digest: 0xc01dcafe
Size: 113,968 bytes
Chunk size: 64 bytes
../163..600/
firmware/

Figure 8.4: Manifest description and fixed-length chunks of a corresponding firmware image.

They include the binary size and message digest as well as parameters for the chunking algorithm. To preserve the authenticity of manifest files, vendors sign them as soon as the firmware executable is generated. This also protects the message digest, which is later used to validate the final firmware image on the IoT devices. The manifest is addressed using the base name (see Figure 8.3) and the suffix */manifest*.

Firmware upload and binary management. Once all artifacts have been produced, a vendor delivers the manifest and firmware chunks to the corresponding deployment operator to serve them in a firmware repository. The publication process runs in an automated manner and requires an authentication framework to ensure consistency and security, such as the publicly auditable bookkeeping service NDN DeLorean [212]. A firmware repository stores versioned images of all vendors and retains them until they are purged. For replication purposes, an operator can deploy multiple firmware repository instances, which then synchronize using any data set synchronization solution [226, 227].

The uploaded firmware binaries and manifests are tagged following the naming scheme in Figure 8.3. The suffix for the actual image is */firmware* and the corresponding manifest is */manifest*; chunks are accessed via */chunk/id*. The timestamp in the naming scheme updates for new firmware versions to reflect the upload time and the granularity of the epoch time is coordinated with the polling interval of the devices, *e.g.,* a daily alignment on midnight would yield `1632261600` for `09/22/2021 00:00:00`. A vendor chooses different degrees of granularity on a device class level as illustrated in Figure 8.5.

Discussion: full versus incremental updates. We design our firmware roll-out approach to always deliver the full binary. An alternate approach would explore the use of differential algorithms to compute software differences, *e.g.,* with *bsdiff* [228], and transmit them in the form of binary patches. While it is undemanding for powerful firmware repositories to calculate a minimal diff representation, the patch size can grow very quickly for compiled binaries. Especially in the IoT, binaries are compiled with optimizations to reduce the binary size as far as possible to fit the image on the programmable flash memory. This can lead to large differences for small changes between software versions due to code re-organizations, up to the point, where caching them in the content store becomes unfeasible and would evict application data.

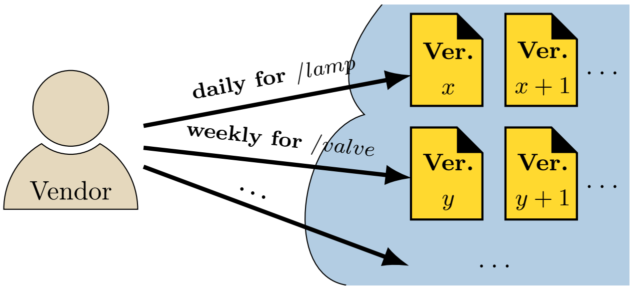


Figure 8.5: Vendor publishes firmwares and aligns date granularity with polling interval of device classes.

Incremental updates using the linear chunking approach is another alternative, which appears to be attractive on first sight. Chunks from previous versions could be reused with a correct name mapping in the manifest file to reorder the fragments independent of their sequential chunk identifier. However, this can quickly inflate the size of the meta-data itself and may require a separate fragmentation for the manifest file.

Sophisticated linking techniques that use auxiliary information about the structure of deployed software modules [229] can produce concise patches compared to naïve diff algorithms which operate on the byte level. Run-time relinking of software components directly on the sensor nodes can lead to minimal diff representations, but requires an extensive tooling support during binary compilation and software installation [230]. While we only focus on the propagation of the binary, the actual representation of such an artifact (full binary or an increment) is rather secondary and may only necessitate slight protocol adaptations regarding the naming schema.

8.2.3 Firmware update process

Firmware version discovery. IoT devices may be provisioned with an up-to-date firmware version before they become operational in a deployment. Over time, vendors release new software versions to update device functionalities or to handle security related issues. Depending on the network availability, a device may be a single or multiple versions behind the current firmware. A version discovery is therefore the first step to any upgrade process.

Two fundamental strategies exist when determining the availability of a new firmware update: (*i*) proactively notifying the IoT devices using push mechanics, and (*ii*) periodically polling the firmware repository. While timely notifications from a firmware server to the IoT device minimize the operational run time of outdated software components, it also bears the following issues. First, notifications are not guaranteed to arrive in low-power regimes where nodes favor extended sleep cycles. Second, it requires server-side state and maintenance overhead to keep track of deployed versions as well as topological information to ensure node reachability, and last, the push mechanism is not native to NDN.

Our approach primarily relies on a pull-driven version discovery, where the embedded devices periodically request the latest manifest file. Vendors convey a sensible polling interval on device class level, *e.g.,* a daily check on midnight for remotely deployed gas valves, or flexible intervals based on harvested energy levels for battery-less sensors. These guidelines are programmed during run-time configurations and may change at an operator's discretion. Since the Unix epoch denotes the actuality of a firmware image in the name schema, all IoT devices need to re-adjust drifting system clocks using an external mechanism, *e.g.,* by relying on the time information of an equipped GPS module, or by operating a time synchronization protocol, such as NDNTP [231].

Retrieval of firmware versions. To discover a new version, IoT devices send Interests to the name that identifies the latest firmware version by setting the correct time frame. Following our previous example, the Interest may describe the name */OilRig-3/IoTCompany-5/Valve-7/1632261600/manifest*, in which the requested time frame is greater than the time frame of the locally running firmware. The firmware repository returns a manifest file if the requested update is available, *i.e.,* a vendor published the binary image for the specified time frame. Interest retransmissions retry the update request for a configurable, but limited number of times to recover manifests from packet loss. In the event that a requested manifest does not exist yet or all corrective actions fail, the Interest times out as part of the default NDN forwarding logic and the IoT node triggers a subsequent update request on the next polling interval, potentially on midnight of the next day. Negative acknowledgments for Interests (NACKs) is a supported NDN protocol element to hint at the absence of requested data or to carry nuanced error codes of the application. For our retrieval mechanism, they may include application-level indications about the latest firmware version. While NACKs are not necessary to ensure a continuous operation, this feature (*i*) provides an optimization to reduce the amount of retransmissions when polling for a new, non-existent firmware version, and (*ii*) assists with the convergence of updates for devices that missed a version publication, *e.g.,* due to network unavailability. The lifetimes of cacheable NACKs need to be aligned with the firmware release cycles to prevent them from wrongly satisfying requests for eventually released versions. To reduce the attack surface, NACKs require similar security considerations as manifest packets.

IoT nodes may be disconnected for longer periods from the core network and thereby may fall several versions behind. Fixing a maximal update frequency at the application level allows a node to always request the latest version at the appropriate Unix epoch. Hence, outdated devices need not attempt to retrieve obsolete versions. Forwarding states are handled by an external routing system, *e.g.,* [208, 44], preferably using a single default route from all IoT devices toward the firmware repository.

Implicit consumption of firmware versions. Polling intervals of devices within the same class can drift apart over time, so we utilize an implicit version discovery process to reduce the amount of individual manifest requests and to increase the reactivity of the firmware roll-out. Each IoT forwarder in a multi-hop request path compares incoming manifest requests with

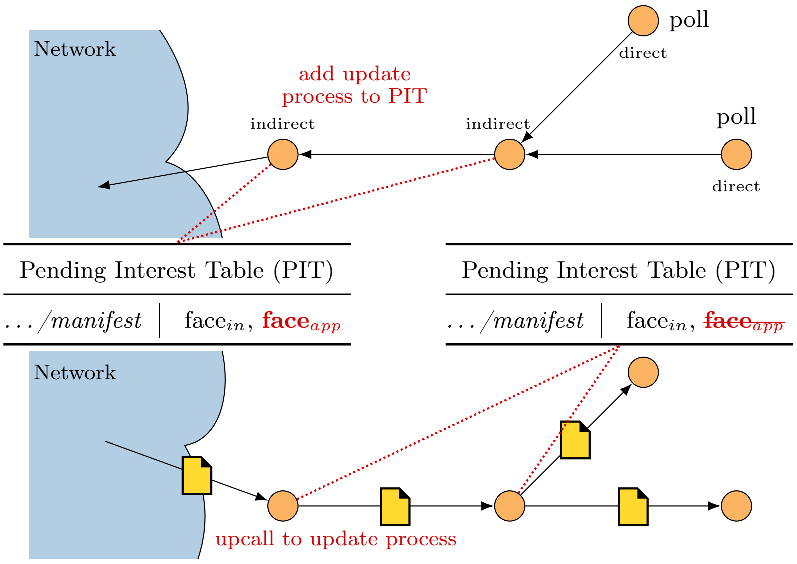


Figure 8.6: Direct and implicit version discovery.

its own device class. On a positive match and if the requested name has a greater epoch time than the currently operating firmware, then the update process of a forwarding device internally registers to the same entry in the Pending Interest Table (PIT) as illustrated in Figure 8.6. This assures that each device of the same class on a request path consumes the manifest and then initiates the retrieval procedure of the firmware image before its local request interval triggers.

Retrieval of firmware image chunks. Once an edge node determines the need for a version upgrade by receiving an up-to-date manifest file, it prepares for retrieving the associated binary image. Initially, the manifest signature is validated using key materials previously provisioned by a vendor. On a failed check, the upgrade process aborts and this incident is reported to the vendor. A valid signature triggers the retrieval of all firmware chunks as designated by the manifest. Each chunk is addressed by appending the chunk identifier (*/chunk/id*) to the base name, where *id* starts at 0 and gradually increments to the maximum chunk number as appointed by the manifest. This also ensures that a few resources are available for alternative forwarding duties. Since memory and network resources are generally limited in low-power regimes, the system uses a stop-and-wait automatic repeat-request (ARQ) error-control method, *i.e.,* each chunk is retrieved iteratively as illustrated in Figure 8.7. Characteristically, resource-constrained class 2 devices [1] equip less than 100 KiB of main memory, where larger parts are inevitably consumed by the operating system, the network stack, and reserved for application purposes. This leaves only persistent memory components, *e.g.,* flash and SD cards, to buffer intermediate chunks during the retrieval. In contrast to the available RAM, external memory often displays storage capacities that are orders of magnitude larger, but uncoordinated access can also consume the available energy budget as I/O operations tend to be slower and energy-draining.

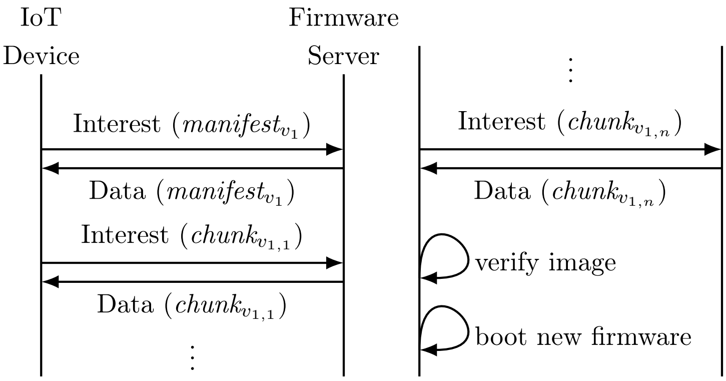


Figure 8.7: Iterative retrieval of firmware chunks.

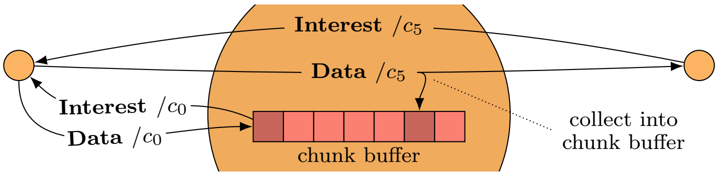


Figure 8.8: Local buffer collects chunks from overlapping upgrade processes.

We define two different chunk retrieval strategies that we assess in our experimental evaluations. The first method allows concurrent firmware updates from nodes on the same request path. The second retrieval method disallows overlapping updates and rather prefers an ordered update that cascades downstream into the IoT network.

Concurrent firmware updates. While nodes request one chunk at a time, they still perform forwarding duties for other devices. Overlapping upgrade processes may also yield incoming data objects that are farther advanced in the firmware buffer than the local chunk identifier as illustrated in Figure 8.8. In this case, a firmware consumer diverts matching chunks with higher progression into the local buffer. Simultaneously, this buffer is also used for serving incoming chunk requests from other devices. Although this optimization results in an unordered data retrieval, the use of fixed-length chunks eliminates the need for reorganizing the fragments when reconstructing the image. Power demanding I/O operations to persistent memory are thus minimized.

Cascading firmware updates. In this retrieval method, a node denies the delivery of firmware chunks for the same device class as long as a node did not complete the update process itself. Downstream nodes run into request timeouts for the first chunk and application retransmissions retry the retrieval using a configurable polling interval. With this strategy, firmware versions propagate hop-wise from a gateway device towards any leaf node of a multi-hop network.

Firmware verification. After completing the retrieval, all necessary chunks reside in the local chunk buffer and this also concludes the full image reassembly. A node calculates a message digest across the buffer and validates it against the previously received firmware digest. On a

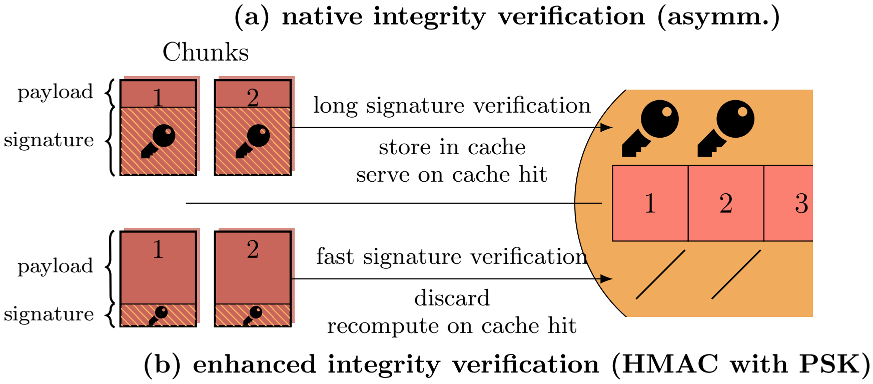

(a) native integrity verification (asymm.)
Chunks
payload
signature
1
2
long signature verification
store in cache
serve on cache hit
1
2
3
payload
signature
1
2
fast signature verification
discard
recompute on cache hit
(b) enhanced integrity verification (HMAC with PSK)

Figure 8.9: Enhanced chunk-wise integrity verification to save device and network resources compared a native NDN protection with asymmetric cryptography.

positive verification, the binary is copied to the correct flash region, the temporary chunk buffer is cleared, and the bootloader is notified to invoke the new firmware. The timer for the next version request is armed as soon as the new image boots successfully. Following the SUIT [213] philosophy, the update process still keeps the old firmware image on the device as a backup in case the recent firmware update breaks the node operation. At worst, the bootloader initiates a fail-safe to re-flash the old binary and return to a correct and consistent behavior.

Firmware replication on connectivity loss. Once the upgrade completes, a device can also serve the latest manifest and binary chunks to downstream devices. The advantage of using a linear binary chunking is that an up-to-date forwarder device serves chunk requests directly from its read-only flash region where the currently running firmware resides, without separately consuming main memory. A firmware version can therefore cascade downstream into the IoT network in a hop-by-hop fashion without necessary operations from the firmware server. This design confines chunk retrievals to a single link and therefore leads to a reduction in bandwidth usage. It also provides a loose coupling, so that upgrade processes become resilient to uplink outages and are unaffected by temporary network disruptions.

Early denial of service (DoS) detection. Images may consist of hundreds or thousands of chunks, depending on the firmware complexity and the (usually small) MTUs of underlying link-layer technologies. NDN protects singular content objects (see Figure 8.9a), but (*i*) the chunk-wise computation of digital signatures using asymmetric cryptography is infeasible for the constrained environment, in particular if no hardware acceleration is available [232]; (*ii*) full-length signatures inflate each packet, thereby immensely reducing the actual goodput of the firmware delivery, and (*iii*) IoT devices must store message signatures alongside the respective data to serve requests from the local cache. This consumes a storage capacity that can grow as large as the firmware itself in low MTU scenarios (*e.g.,* for 802.15.4 with less than 128 bytes payload room).

Figure 8.10 illustrates the aggravating effect of comparatively large signature sizes. In this

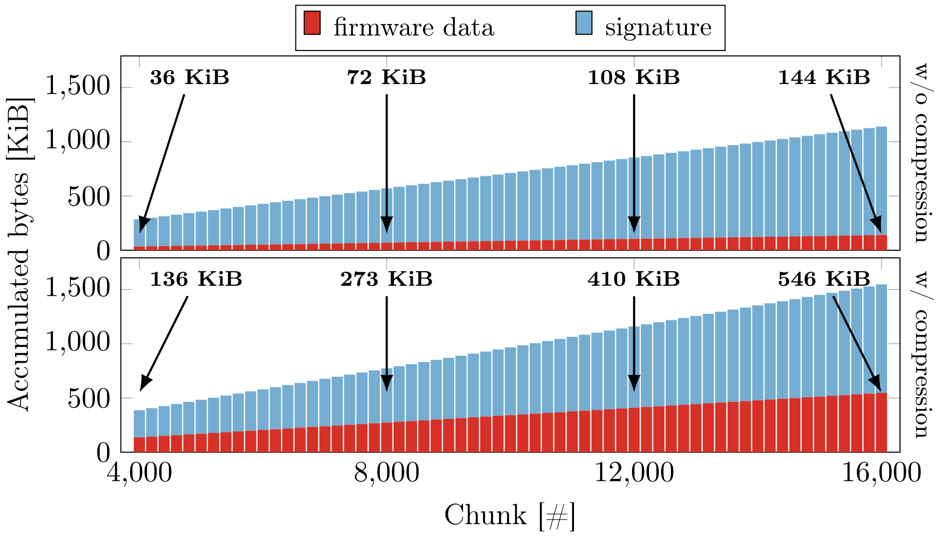


Figure 8.10: Chunk-wise signature overhead compared to the actual firmware data. Chunks contain 9 bytes (w/o ICNLoWPAN compression) and 35 bytes (w/ ICNLoWPAN compression) of application data. Signatures are 64 bytes for EdDSA (Curve25519).

example, we assume the 802.15.4 MTU, a data name of 16 bytes, a structural NDN encoding overhead of another 16 bytes, and the link-layer header further consumes 23 bytes when using the long MAC address mode. This sums up to 55 bytes and leaves 73 bytes for the payload and signature. The Edwards-Curve Digital Signature Algorithm (EdDSA) [233] is a prominent choice in the IoT as it provides a high performance and relatively small signatures of 64 bytes—at least with the Ed25519 curve. Yet this reduces the available space for application data down to 9 bytes, resulting in numerous chunk packets containing individual signatures. Even for small firmware sizes of 36 KiB, 4000 chunk transmissions accumulate to a signature overhead of 256 KiB. For larger images, this linearly increases: a firmware with 144 KiB requires 16000 chunk transmissions and produce a signature overhead of 1 MiB. The ICNLoWPAN convergence layer [31] can remove names from Data messages and reduces the structural header overhead. Following our exercise, these enhancements increase the available space for firmware data from 9 to 35 bytes, thereby requiring four times less chunks to complete the firmware delivery. Regardless, the signature overhead remains intolerable. The severity shows in NDN cache environments where each signature has to be stored alongside the chunk data due to the asymmetric aspect of this signature algorithm that prevents IoT devices from generating them.

The integrity and authenticity of a firmware image is validated against the protected message digest from the corresponding manifest file once all chunks have been received and reassembled. Hence, signatures of individual NDN messages are redundant and we omit for the sake of efficiency. Unauthenticated packets, though, open a forceful attack vector to exhaust the resources of the IoT network: Injecting (even few) illegitimate chunks violates the integrity of the firmware and an identification of these invalid chunks is difficult after firmware reassembly. The only approach to recover the binary is then to repeatedly request the firmware, which requires all chunks to traverse the network first. To save device and network resources, a detection of erroneous deliveries and an early exit of the retrieval process is desired.

We augment individual chunks with a keyed-hash message authentication code (HMAC [234]) that is verified upon reception (see Figure 8.9b). Next to the asymmetric cryptography, NDN already provides the protocol elements to encode a 32-byte HMAC authentication code. To check for data integrity as well as authenticity, the HMAC requires seeding. For this early DoS detection module, we assume a pre-shared secret at all devices of a class, which can be pre-installed by the vendor or obtained in an out-of-band manner and eventually protected in secure memory. A chunk is then recorded in the chunk buffer only after correctly verified by its recipient. If a chunk validation fails, a recipient repeats requests for invalid chunks only. After iterated (*e.g.,* three) failing verification attempts, a node marks the firmware as irrecoverable, aborts the update process, and notifies the vendor.

It is noteworthy that these signature hashes based on pre-shared secrets can be discarded during caching in the chunk buffer, since nodes of the same device class can easily re-generate them using the same secret at any time. This relieves storage capacities, while preserving an intact cache operation for incoming chunk requests. For low-power regimes with small-sized MTUs, a full HMAC signature may occupy too many bytes in a frame. For optimization, we only transmit a configurable prefix of the hash, *e.g.,* 8, or 16 bytes. This trade-off increases the susceptibility to hash collisions, but drastically increases the goodput. Security and robustness of the final image verification remain unaffected by these optimizations.

8.3 Experimental Evaluation

In this section, we quantitatively assess our previously outlined information-centric firmware update approach using a real protocol implementation and constrained nodes in a testbed.

8.3.1 Experiment setup

Scenario and network topology. We conduct our experiments in a wirelessly connected IoT deployment where a gateway node is situated at the network edge to provide an uplink connectivity to a set of 30 IoT devices. A new binary version is rolled out into the stub network. On system initialization, the constrained nodes statically arrange in a destination-oriented, directed and acyclic graph (DODAG) as depicted in Figure 8.11. DODAG topologies provide shortest paths from IoT devices to root nodes (*i.e.,* gateway or cloud) and therefore incur a minimal routing overhead for the prevalent *converge cast* scenario, *i.e.,* a large amount of traffic is directed to or from a central point. In fact, RPL [202]—the predominant routing protocol for the IoT—uses DODAGs as a fundamental part of its routing system. While we rely on a static topology in our test environment to sidestep the delays of routing convergence and

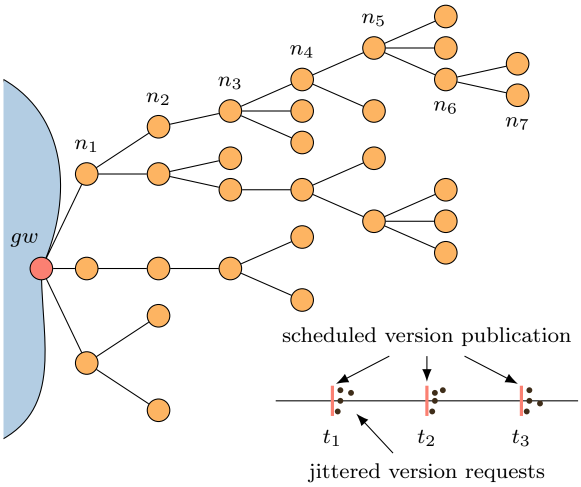


Figure 8.11: Logical testbed topology modeling multiple branches from rank zero to seven of the forwarding hierarchy.

to solely focus on the propagation of large data objects, an authentic deployment would use an orthogonal routing protocol to dynamically construct and repair the DODAG as necessary.

Software and hardware platform. On all IoT nodes, we deploy the RIOT [28] operating system in version 2021.04. It integrates with CCN-lite, which implements a minimal NDN forwarder. The necessary update logic runs as a small IoT application using the portability layers of RIOT and CCN-lite, which opens the implementation to a wide range of hardware platforms.

We conduct all evaluations on the FIT IoT-LAB [27] testbed. It features large deployments of several ARM Cortex-M3 based class 2 devices [1] with 64 kB of RAM and 256 kB of ROM. The testbed nodes are equipped with an Atmel AT86RF231 [69] transceiver to operate on the IEEE 802.15.4 2.4 GHz radio.

Deployment parameters. We externally align the system clock of the IoT devices and the gateway node with the Unix epoch using the instrumentation tools of the testbed. In a configured interval of one hour, we generate new binary versions and record the corresponding manifest and image files in the content store of the gateway. Once the time is synchronized, the IoT nodes request new manifest files from the gateway node as soon as they are generated. In our experiment, we deploy the same device class throughout the network, *i.e.,* the same firmware image for all devices, but also provide a glance at the end of the evaluation on the performance for the other extreme: all nodes are of a different device class. We separately explore the two retrieval strategies: *concurrent*, where update processes overlap among multiple nodes, and *cascading*, where downstream nodes first wait for upstream nodes to complete the update.

We choose names for manifests and chunks (see Figure 8.3) with a total size of 45 bytes when

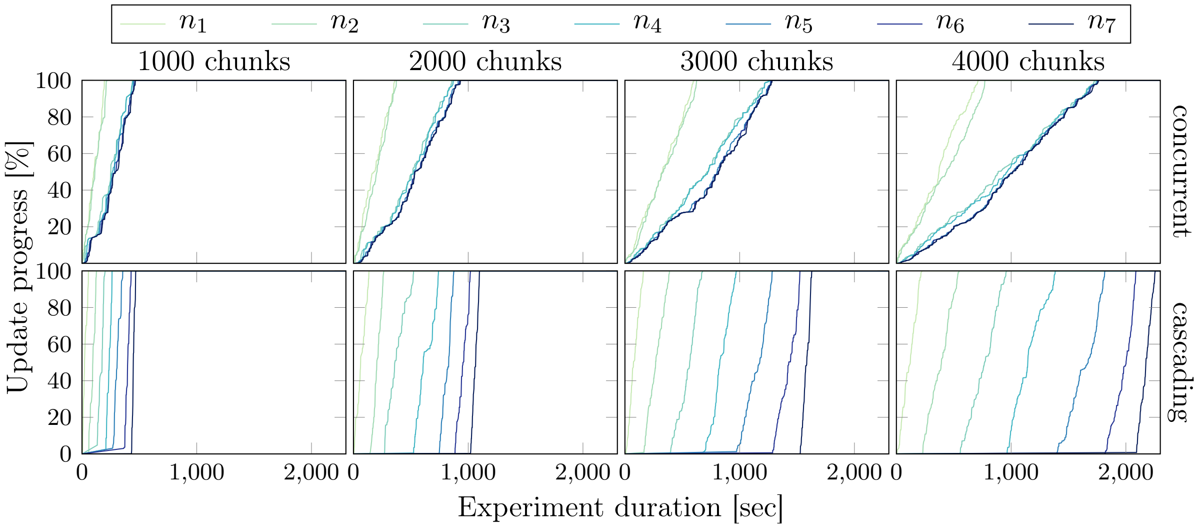


Figure 8.12: Overall firmware update progression for the selected nodes $n_{1...7}$ with an increasing number of maximum chunks using the *concurrent* and *cascading* retrieval strategies.

encoded in the NDN TLV format. While we increase the image size from 32 to 128 kB in our experimental evaluations, we gradually raise the number of maximum chunks from 1000 to 4000. Thereby, we keep the chunk size fixed to 32 bytes across all configurations. This yields a length of 92 bytes for chunk data packets and the total frame size sums up to 115 bytes including the IEEE 802.15.4 link header. Thus, these parameters produce chunk packets that are very close to the link MTU of 128 bytes. The NDN forwarder performs three retransmissions in a two-second interval and the application triggers retransmissions in a jittered interval of 10±5 seconds after a designated chunk request times out. We configure three link-layer retransmissions that operate in the lower millisecond range, whereby each retransmission is slightly delayed by a random exponential backoff algorithm.

8.3.2 Firmware update progress

In our first evaluation, we gauge the update progression over time for a set of selected nodes with increasing firmware size. This nodal time measurement starts when the first firmware chunk is requested and terminates on the successful delivery of the last chunk. Our node selection consists of $n_{1...7}$, *i.e.,* the nodes that reside on the longest path in our topology. Figure 8.12 summarizes the various evolutions over the experiment duration.

We observe that both retrieval strategies yield very different progression charts. In the *concurrent* mode, all update procedures of $n_{1...7}$ start almost simultaneously and run concurrently for a designated time. The first two nodes $n_{1,2}$ advance with a similar chunk retrieval speed in all configurations and the remaining nodes $n_{3...7}$ display a similar alignment, albeit with a much slower evolution. While the firmware distribution with 1000 chunks continues for ≈8 minutes to

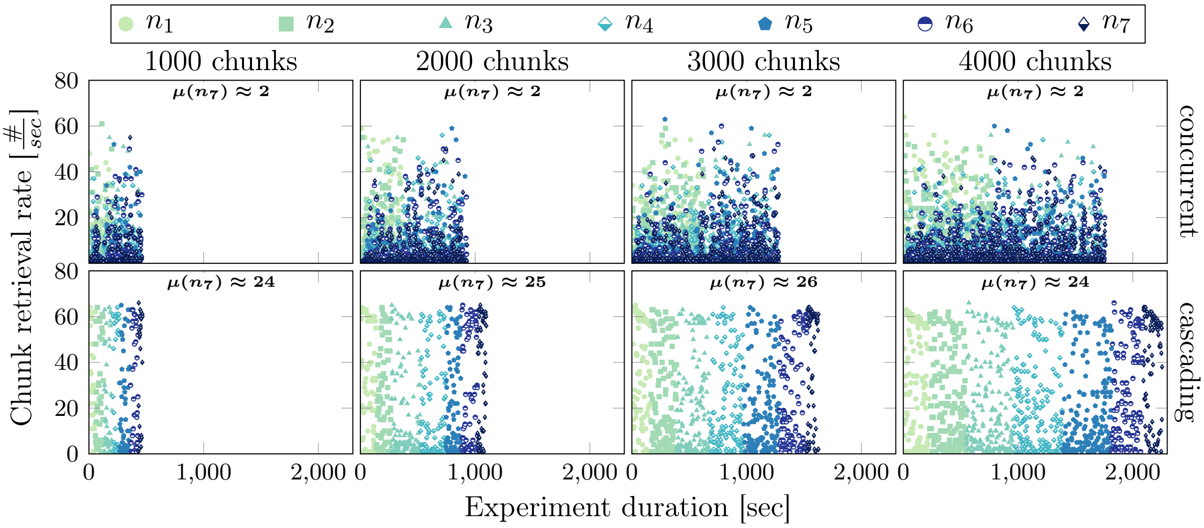


Figure 8.13: Chunk retrieval rate per second for our node selection using both retrieval strategies.

complete for the whole network branch, the duration increases to ≈30 minutes for an image file of 128 kBytes (4000 chunks). The *cascading* deployments display the anticipated stop-and-wait characteristic. Single nodes wait for the immediate uplink node to finish the update process, before any chunk retrievals are invoked. This serialization positively affects individual update speeds. In the extreme configuration, the update duration for n_7 declines from 30 minutes down to 3 minutes, which is the quickest update completion on the request path. However, while individual updates appear to be faster in the *cascading* mode, the global roll-out on this path is ≈8 minutes slower than in the *concurrent* mode.

8.3.3 Goodput analysis

In our next comparison, we emphasize on nodal chunk retrieval rates to elucidate the previous progression differences. Figure 8.13 displays the amount of accumulated chunks that nodes retrieve in a second. We observe highly fluctuating rates throughout the update process ranging from zero chunks per second up to accumulated retrievals around 60 chunks per second. With the *concurrent* retrieval strategy, nodes $n_{3...7}$ generally display lower rates while $n_{1,2}$ have ongoing transmissions. The average performance of the n_7 leaf node nets to an average of approximately 2 chunks per second for all configurations. Roughly at the middle of the experiment duration the first two nodes complete their update process, which leads to slightly increased retrieval rates for the remaining nodes. This is an indication that nodes in this deployment are competing for bandwidth in the shared wireless medium. When retrievals are *cascading*, then the number of simultaneously competing nodes in the topology is drastically reduced to single nodes in all request paths of the topology that have overlapping broadcast ranges. The nodal goodput

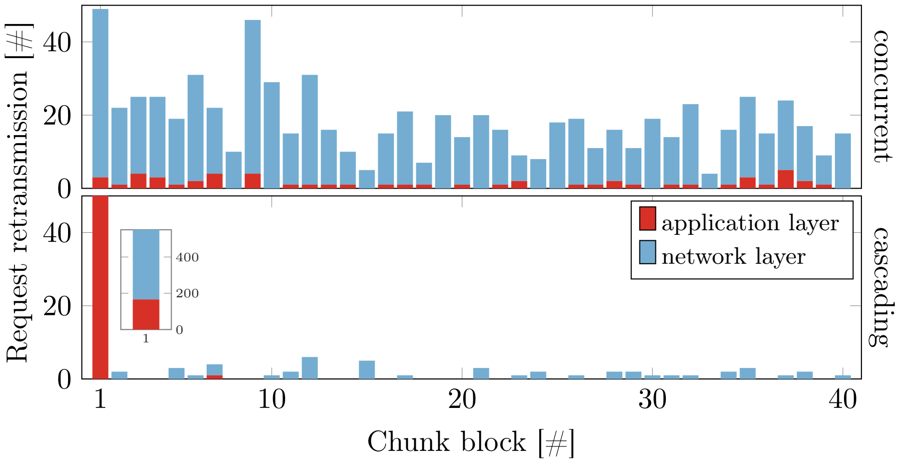


Figure 8.14: Chunk request retransmissions on the application and network layer grouped into blocks of 100 chunks for the n_7 node.

moderately improves for all nodes across all presented configurations. For n_7, this translates into a performance gain that is nearly twelve-fold. The evident oscillations are a result of request retransmissions. Unlike layer 2 retransmissions which operate on the millisecond range and are mostly invisible in the considered timescale, corrective actions on upper layers block the retrieval process by multiple seconds until messages are recovered by the network layer or application, thereby impairing the nodal goodput rates.

8.3.4 Link stress

The preceding evaluations suggest that both retrieval methods experience varying degrees of network stress when firmware updates are progressing in the multi-hop topology. We now measure the link stress for n_7 by quantifying the amount of retransmitted chunk requests. Figure 8.14 accumulates request retransmissions for blocks of 100 chunks and differentiates between corrective actions on the network and application layer. In the *concurrent* configuration, n_7 triggers a seemingly continuous stream of ≈5–45 retransmissions which is higher at the beginning and then slightly decreases over the experiment duration. This is in accordance with our former observation that chunk rates increase as soon as competing upstream nodes complete their updates and access to the shared medium lessens. Overall, the number of application retransmits is minuscule compared to the number of network retransmissions, *i.e.,* NDN is able to recover most of the chunks with its three request attempts.

The *cascading* setup shows a much less pronounced retransmission behavior: many chunks experience no packet loss at all while other groups register fewer than ten network retransmissions for 100 chunks—still considerably less than the *concurrent* configuration. This relaxed progression also confirms the previously observed performance gains when firmware images are distributed in a hop-wise fashion. Application retransmissions are virtually absent, excluding

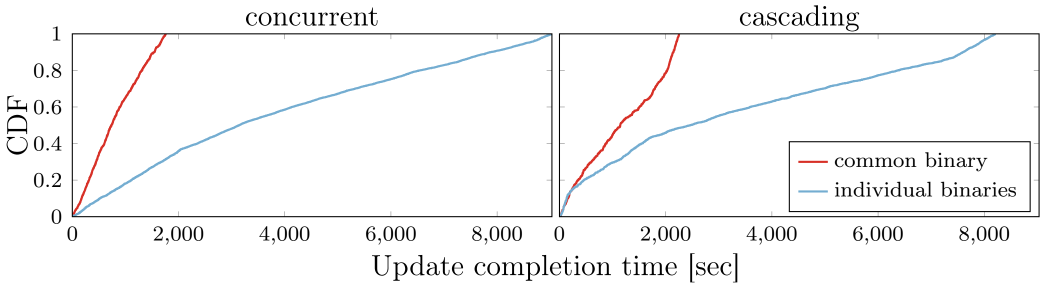


Figure 8.15: Update completion time for the selected path $n_{1...7}$ with a maximum amount of 4000 chunks per firmware image.

the very first chunk. n_7 retries the retrieval of the first chunk, but n_6 denies the delivery until it completes its own update. This leads to the large number of ≈160 application and ≈500 network retransmissions that originate from n_7. These numbers appear to be disproportionately high, however, these packets trigger moderately in the seconds range over a period of ≈30 minutes and do not pose a significant stress to the shared medium. Overall, we observe sufficient idle resources to continue regular network operations during the update.

8.3.5 Multiparty assessment

Up until now, our experiments updates the same device class throughout the network. A roll-out of a common firmware image clearly benefits from the NDN multicast support: in-network caches and request aggregations can greatly balance the network utilization. In this last assessment, we configure a different device class for each device in the deployment to deliver individual binaries to the respective nodes. While this contrary extreme is usually impracticable in real-world deployments, it gives a sensible estimation on the performance of protocol ensembles without caching and aggregation capabilities. Due to the low memory, nodes are only able to cache a maximum of 64 foreign chunks, but they mostly evict before they can be utilized by retransmissions, because of the significant chunk flow rate that leads to rapid and frequent cache replacements. The internal chunk buffer is reserved for the respective binary image of the node and is therefore inaccessible by the remaining nodes.

We measure the chunk arrival times for nodes $n_{1...7}$ and demonstrate the update progression in Figure 8.15. Nodes request 4000 chunks to complete the image delivery, *i.e.,* 28k distinct chunks in total are transmitted on that particular path. The distributions indicate a completion time of ≈30 minutes for the setup with a single device class and a common binary. On the other hand, the update time considerably decelerates if the NDN multicast features are inactive. Hence, the update process continues for more than two hours when individual binaries are deployed to propagate. The missing hop-wise caching ability means that retransmissions need to traverse the full request path up to the gateway node on each retry, which again promotes higher packet loss probabilities due to the generated side traffic for other, ongoing transmissions. In contrast, in-network caches reduce the number of necessary hops and confine retransmissions in the best case to a single link. The greater slopes towards the end of both *cascading* measurements are an indication that leaf nodes operate quicker with the coordinated retrieval method due to absent nodes in the vicinity that compete for the bandwidth, irrespective of caching abilities.

8.4 Conclusions

We have studied massive roll-outs of firmware in large-scale constrained multi-hop networks, which is an emerging need but also a major challenge for the IoT edge. We found that information-centric content replication fosters efficient and reliable chunk dissemination, which makes routine firmware updates feasible even for nodes that are highly constrained in processing power, memory, and radio capacity. Hop-wise forwarding and in-network caching in particular facilitate update campaigns across homogeneous wireless regimes even with intermittent connectivity.

Using the IETF SUIT update model as a blueprint, we further devised and evaluated firmware propagation strategies based on the Named Data Networking (NDN) protocol. We conducted a feasibility analysis using real protocol implementations on a wireless testbed to quantify the effective network performance of retrieving large firmware images in the information-centric Internet of Things. Our findings indicate that (*i*) a simultaneous, uncoordinated distribution of firmwares results in high cross traffic within the broadcast domain that degrades nodal operability, (*ii*) deployments with common binaries significantly benefit from in-network caching, and (*iii*) a hop-wise, cascading delivery relaxes strain on network resources, allows for continued regular operations during the roll-out process, and preserves limited energy budgets by allowing longer sleep cycles due to prompt firmware deliveries.

Chapter 9

Analysis of Content Object Security for Constrained Regimes

Abstract

Massive amounts of content objects are published and exchanged every day on the Internet. The emerging Internet of Things (IoT) augments the network edge with reading sensors and controlling actuators that comprise machine-to-machine communication using small data objects. IoT content objects can often be sent in messages that fit into single IPv6 datagrams. These IoT messages frequently traverse protocol translators at gateways, which break end-to-end transport and security of Internet protocols. To preserve content security from end to end via gateways and proxies, the IETF recently developed Object Security for Constrained RESTful Environments (OSCORE), which extends the Constrained Application Protocol (CoAP) with content object security features commonly known from Information Centric Networking (ICN).

This chapter revisits the current IoT protocol architectures and presents a comparative analysis of protocol stacks that protect request-response transactions. We discuss features and limitations of the different protocols and analyze emerging functional extensions. We measure the protocol performance of CoAP over Datagram Transport Layer Security (DTLS), OSCORE, and the information-centric Named Data Networking (NDN) protocol on a large-scale IoT testbed in single- and multi-hop scenarios. Our findings indicate that (a) OSCORE improves on CoAP over DTLS in error-prone wireless regimes due to omitting the overhead of maintaining security sessions at endpoints, (b) NDN attains superior robustness and reliability due to its intrinsic network caches and hop-wise retransmissions, and (c) OSCORE/CoAP offers room for improvement and optimization in multiple directions.

9.1 The Problem of Securing IoT Content and Related Protocol Work

9.1.1 Problem Statement

The Internet of Things is evolving to connect numerous, often constrained devices that regularly exchange massive amounts of data. Authenticity and possibly confidentiality of information is of vital interest in a wide range of applications. The problem, though, is that low-end devices need to optimize scarce resources and thus need to minimize cryptographic operations and state while (re-)transmitting packets [235].

At the same time, low-power lossy networks frequently experience packet loss and require retransmissions—multihop transfers often significantly challenge these error-prone regimes [236]. Overhead from cryptographic credentials or signaling security sessions consumes additional energy and may quickly become critical for these low-power devices.

Low-end IoT nodes often operate intermittently to save resources [161]. Duty cycling or energy shortages may force devices into deep sleep with little capacities for saving protocol state or security credentials in non-volatile memory modules. Firmware updates, disruptive environments, or intermittent power availability may repeatedly cause unanticipated system resets. Any of these harsh conditions may lead to a loss of state at endpoints. Once lost, session and security state needs reestablishing to continue operation. Methods for replicating protected data, as well as lightweight recovery mechanisms from state loss including an efficient rediscovery and re-association of networked nodes are hence vital for seamless, perpetual operations: they save computational resources, radio cycles, and preserve system energy.

Many IoT scenarios such as multi-destination control messaging or convergecast sensor readings, but in particular over the air (OTA) firmware updates can take advantage of multi-party communication, which in the wireless IoT often pairs with mobility. Multicast mobility is an asymmetric problem [237]. While the movement of receivers is often easy to compensate through local network reconfigurations, the impact of mobile sources on routing and forwarding is complex and requires assisting measures or services. Secure communication contexts require proper group keying and secure group membership management, which require dedicated treatment on the protocol level if bound to communication endpoints.

In a wider context, trust relationships in the IoT are heterogeneous and change with varying deployment settings. While the exchanging endpoints are often widely distributed (*e.g.,* sensors and a cloud), IoT gateways often need to translate between protocols. If translators are required to re-authenticate and re-encrypt, all communicating parties must pre-establish trust with the IoT gateways in place. This is likely to be a major problem in provider-bound deployments such as 5G.

There are several ways to securely transfer content across the constrained IoT. The most common approach builds on securing the transport layer, which establishes confidentiality and trust between end points and remains neutral with respect to application layer protocols. An option for content object security has been newly developed for CoAP communication between IoT nodes, after a longer research period on information-centric IoT networking had worked out the advantages of securing content objects autonomously. We discuss properties, underlying mechanisms, and protocols for these three approaches in the following three subsections.

9.1.2 Transport Layer Security in the IoT

Datagram Transport Layer Security (DTLS) [25] closely follows TLS [60] in terms of protocol behavior and security guarantees. Unlike its stream-oriented relative, DTLS adds facilities to operate in unreliable datagram environments. It contributes a modified record layer that tolerates packet loss and message reordering. To break up inter-record dependencies, DTLS bans stream ciphers and uses explicit sequence numbers in datagrams. A cryptographic context thus spans exactly one record. UDP is the prevailing transport in IoT deployments. Compared to TCP, it exhibits no substantial protocol overhead and allows for implementations with low memory footprints. Utilizing DTLS to secure existing application protocols such as CoAP and MQTT-SN hence appears to be the best logical choice—at least at first glance.

Concerns arose in recent studies that question the applicability of DTLS in large-scale IoT systems. First, certain cryptographic challenges during handshake processes are infeasible. While processing time for cryptographic operations diminish with hardware crypto modules, message sizes are inflated. Asymmetric key ciphers require handshake overhead and large payload sizes, which immensely boost handshake completion times to the order of seconds and minutes in multi-hop deployments due to packet fragmentation [238].

The stateful session characteristic further comes at the cost of multicast capability, since security contexts are identified by the classic 5-tuple between two endpoints. Particularly in scenarios that involve device mobility and multi-homing, a generally accepted effort applies connection identifiers to security channels—independent of the 5-tuple [239]. Figure 9.1 (a) illustrates a realistic deployment setup for CoAP over DTLS: End-to-end security commonly terminates at the gateway to allow for protocol conversions, *e.g.,* to HTTPS over TCP.

9.1.3 Content Object Security in the IoT

OSCORE [26] is a protocol extension to CoAP and addresses the terminating security issue at gateways. Instead of securing sessions between endpoints, OSCORE protects entire CoAP messages and provides integrity, authenticity, and confidentiality on an object level. The original CoAP message is thereby encapsulated as an authenticated and encrypted CBOR Object Signing and Encryption (COSE) [240] object by an outer CoAP option. In addition to cryptographic efforts, the protocol further includes countermeasures to prevent response delay and mismatch attacks. A strong message binding between requests and corresponding responses is constructed with the use of identical identifiers in their authenticated components, which

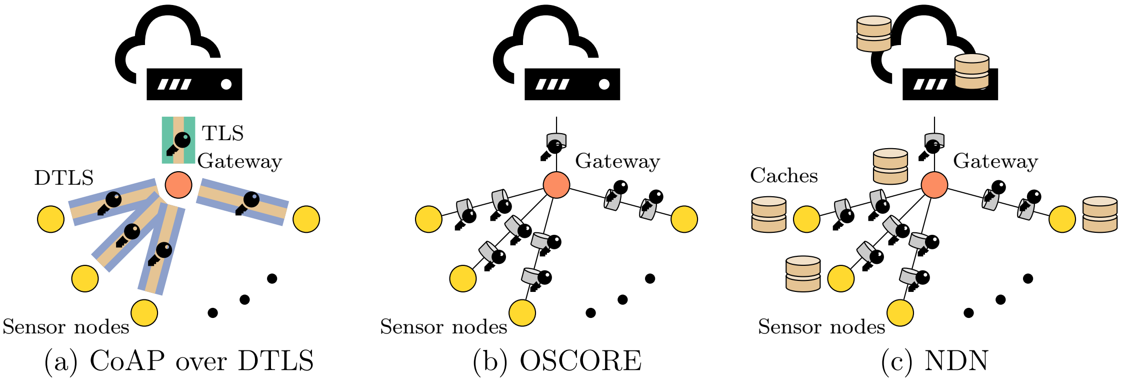


Figure 9.1: Typical deployment setups for CoAP over DTLS, OSCORE, and NDN in the IoT. Validity of session keys terminates at gateways for transport layer security due to transport conversions, *e.g.,* UDP to TCP. Content object security is unaffected by gateway operations and reaches end-to-end.

persist over retransmissions. Replay windows allow for rearranged messages to be processed independently. Applications built on it use CoAP mechanisms like If-Match or the Echo [241] option to protect against any ill-effects of rearranged messages.

OSCORE utilizes the request-response semantics of its underlying CoAP layer and an elaborate nonce construction to obtain compact response messages. When combined with CoAP observation (continuous responses to a single request), OSCORE protects the sequence of notifications using its own sequence numbers. When combined with CoAP block-wise transfer, it fragments large resources into pieces small enough for the end points to process in a single cryptographic operation without hindering further block-wise processing by proxies. Unlike DTLS, OSCORE does not come with a built-in key exchange protocol, and relies on pre-shared keys. A lightweight authenticated key exchange (LAKE [242]) is under development as a companion protocol.

A major improvement over the conventional transport layer security concept is the ability to secure multicast messages. CoAP supports a one-to-many group communication [243] when used with UDP. While DTLS fails to perform in multicast environments, the object security characteristic of OSCORE allows for protected requests and responses in these deployments [244].

Figure 9.1 (b) illustrates the envisioned deployment option. Messages are cryptographically secured and despite protocol conversions on gateways, their properties stay intact while traversing them forward up to cloud services.

9.1.4 Content Security in the Information-Centric IoT

Information-centric Networking [11, 12]—a clean-slate approach of the Future Internet initiatives—abandons the host-centric Internet paradigm in the favor of autonomous content, which allow

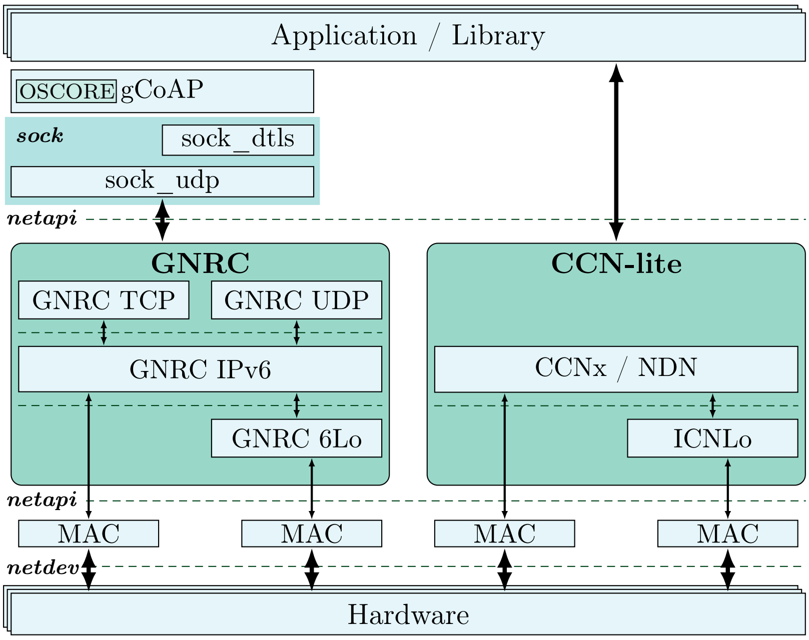


Figure 9.2: The RIOT networking subsystem.

for an unhindered replication of authenticated data objects as illustrated in Figure 9.1 (c) and further described in Section 1.2. Named Data Networking (NDN) [21] supports integrity and authenticity as protocol features by appending cryptographic signatures to data packets. While originally the intrinsic security only applied to data packets, the upcoming NDN protocol version allows for a signature inclusion in Interests. Confidentiality is not supported on the protocol level, but left to the applications, which may encrypt content.

9.2 Composable Network Stacks for Object Security in Challenged IoT Deployments

The decision for a software platform that can cope with constrained IoT is crucial. As we aim for maintainability and sustainability, we extend existing code bases instead of designing and implementing from scratch. As such, we utilize the open source IoT operating system RIOT [28] and leverage its existing network stack, which follows a cleanly layered architecture [29]. In course of our evaluations, we contribute and upstream improvements to the RIOT integrations of DTLS, OSCORE, and NDN.

9.2.1 The RIOT Networking Subsystem

The RIOT networking subsystem displays two interfaces to its externals (see Figure 9.2): The application programming interface `sock` and the device driver API `netdev`. Internal to stacks, protocol layers interact via the unified interface `netapi`, thereby defining a recursive layering of a single concept that enables interaction between various building blocks: 6lo with media access control (MAC), IP with routing protocols, transport layers with application protocols, *etc.* This grants enhanced flexibility for network devices that come with stacks integrated at different levels.

9.2.2 CoAP Over DTLS

gCoAP is the feature-rich native CoAP implementation of RIOT. It implements the server-side and client-side, it supports the most common methods GET, POST, PUT, DELETE, it handles confirmable messages, and it allows for observing resources. gCoAP further provides the CoAP proxy functionality to redirect client requests and maintains a response cache to reduce round-trip times. The cache is using a least recently used eviction strategy to prioritize responses of the most recent requests. As depicted in Figure 9.2, gCoAP uses the sock API. On the north-bound it attaches to `sock_udp` and `sock_dtls`, which makes it completely network stack agnostic. In the default configuration, the native 6LoWPAN [3] network stack of RIOT—Generic (GNRC)—provides the south-bound implementation of sock_udp. The DTLS counterpart is provided by the external package tinyDTLS. It follows a threadless design and depends on events, which are handled by the sock layer within the gCoAP thread. This DTLS setup supports two ciphersuits: (*i*) pre-shared authentication and key exchange using the Advanced Encryption Standard (AES) in Counter with CBC-MAC mode (CCM) [245] with a 128-bit key length, and (*ii*) an Elliptic Curve Cryptography (ECC) based AES-CCM with an Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) key exchange.

9.2.3 CoAP With OSCORE

We provide `libOSCORE`[1] as an implementation of the OSCORE [26] model that integrates into RIOT. Unlike other approaches, *e.g.,* a yet to be mainstreamed Contiki implementation[2] and c_OSCORE[3] on top of Zephyr, libOSCORE focuses on portability across different CoAP libraries and provides replay protection.

Distinct features of libOSCORE are its handling of the request-response correlation data and its zero-copy approach. In the former, initialization vectors (`Partial IVs`) [240], which are used to create unique nonces for the underlying authenticated encryption algorithm, are consistently passed by reference. They carry a flag indicating first use and get invalidated by consumers. This allows leveraging OSCORE optimizations for safe representational state transfer (REST) operations. In memory management, libOSCORE expects its user to provide suitable memory locations and provides struct definitions to make that portable. This saves execution time, main memory (RAM) and storage (ROM) at the cost of some implementation complexity on

[1] `https://gitlab.com/oscore/liboscore`

[2] `https://github.com/Gunzter/contiki-ng/tree/oscore_12`

[3] `https://github.com/Fraunhofer-AISEC/c_OSCORE`

the user side. It allows processing of messages from the receive-buffer in a single reading pass after in-place decryption and without the need for dynamic memory allocation.

In CoAP libraries that build and read their messages in buffers (*c.f.,* gCoAP), integration of libOSCORE happens in two stages: (*i*) basic integration, and (*ii*) full integration.

The basic integration describes the most elementary way of interacting with libOSCORE. It only requires a mapping of certain CoAP operations and cryptographic primitives. Applications that use this mode directly access OSCORE objects and steer every step of the encryption and decryption process for each packet. Generally, usage of this mode is tedious and error-prone and therefore discouraged for user applications. On the other hand, a basic integration allows for a full control of OSCORE internals, which can be leveraged to perform protocol optimizations by libraries or protocol extensions.

The full integration requires a functional basic integration as prerequisite. At that stage, libOSCORE messages are used as backends for the native CoAP library. Application code is identical for the unprotected and OSCORE-protected case, and thus tightly coupled to the native CoAP implementation. The CoAP library dispatches operations on messages through libOSCORE atop of, or directly through the transport protocol, depending on the application's configuration (*e.g.,* the choice of a security context for a message) or presence of the OSCORE option. This mode of operation is the recommended way of building user applications, as APIs hide security operations and prevent security breaches due to a misuse of OSCORE internals. The stack in Figure 9.2 shows the full integration state where applications interact only with gCoAP.

An intermediate integration is available in libOSCORE for cases when full integration is unfeasible with a particular library or simply incomplete. At that level, an additional library provides code to orchestrate and simplify cryptographic procedures. This mode is most suited for narrow-purpose helper libraries up to full-fledged REST frameworks, which generally provide their own APIs towards user applications.

We further implement experimental protocol features to explore the design space as discussed in our comparative evaluation (Section 9.5). In detail, we allow proxy nodes to cache OSCORE responses so they can be served for request retransmissions from the same client.

9.2.4 Named Data Networking

CCN-lite [106] is a lightweight NDN forwarder, which supports all primary features: in-network caches, hop-wise retransmissions, request aggregation along paths, and multi-source, multi-destination forwarding. It runs on a variety of hardware platforms—ranging from commodity hardware to embedded devices. While the core forwarder is self-contained and platform independent, adaptors provide access to the system communication API. CCN-lite is integrated into RIOT as an external package. It contributes a RIOT adaptor, which hosts its own thread and translates between CCN-lite messages and netapi packets.

9.3 Theoretical Evaluation of Protocol Security

Performance measures such as security properties largely differ for each protocol configuration. In the following, our performance assessment considers protocol design choices and thus provides insights that are independent of specific deployments. We focus on four protocol compositions: (*i*) CoAP (Protected) with encrypted and authenticated response payload as baseline implementation. (*ii*) CoAP over a secured DTLS 1.2 session. (*iii*) OSCORE to provide object security for request and response messages. (*iv*) NDN (Protected) using signed Data messages and encrypted as well as authenticated content.

9.3.1 Cryptographic Algorithms

Cryptographic primitives quickly approach critical resource limits in low-power, wirelessly connected regimes, as computational complexity and memory consumption strongly vary with algorithms and implementations. The limited amount of main memory and slow CPU clock speeds in embedded IoT devices can push completion times of resource-exhaustive operations by orders of magnitude beyond a reasonable performance on commodity hardware—notably for asymmetric cryptography. While long computations reduce sleep cycles and drain batteries much faster, they also affect the responsiveness of the overall—commonly single-threaded—system. Hardware-assisted cryptography can largely improve on resource consumption, but necessary crypto μchips are not always integrated into hardware platforms due to economical reasons. For a detailed analysis of crypto-primitives on low-end IoT devices we refer to [232].

In our following protocol selection, we specifically focused on low-complexity modes in crypto using pre-shared keys to not burden the protocol assessment with disproportionally long intra-stack delays. Since CoAP appoints an authenticated encryption using AES with a block size of 128 bits in CCM mode and an 8-byte authentication tag as mandatory-to-implement [4, Section 9.1.3.1], we configured this choice for all protocol deployments.

9.3.2 Security Properties

CoAP (Protected) exhibits the weakest security properties in our comparison: While it uses an authenticated encryption for the *payload*, it does not provide any security measures for the actual CoAP messages to protect CoAP signaling. Protocol headers are prone to tampering and messages are susceptible to interception as well as packet delay attacks. These shortcomings make the binding of requests to correct responses fragile. The inability to map responses to particular requests is especially dangerous in cases when resources publish mutable content [241, 246]. Consequently, even in the case when the payload is secured, delayed and replayed messages can affect the state machine on the client and server. Since the message headers are not protected against confidentiality attacks, this configuration easily leads to privacy concerns. Plaintext requests will contain resource URIs, which typically help to identify sensitive application information and therefore potentially leak private data. Responses may not include resource URIs, but included tokens unambiguously identify potentially intercepted requests and thus their resource URIs.

CoAP over DTLS is the most common method for securing message transmissions in an IoT network. DTLS provides integrity, authenticity, and confidentiality for UDP datagrams within sessions based on pre-established private keys. It operates below the application layer and inherently takes CoAP requests as well as responses into consideration. A drawback from this layering, however, is that the DTLS record layer is not aware of CoAP semantics. This introduces a twofold problem: First, this configuration suffers from the same request-response binding issues when messages are delayed and replayed [246] while recent mitigations [241] are not deployed yet. Second, end-to-end security terminates at gateways in usual IoT setups when protocol conversions from CoAP to HTTP take place. Minimal DTLS implementations commonly provide the lightweight DTLS cipher suite TLS_PSK_WITH_AES_128_CCM_8 [247], which does not provide perfect forward secrecy. Adaptations [248], allow for the combination of existing cipher suites with the Ephemeral Elliptic Curve Diffie-Hellman key agreement protocol.

OSCORE achieves a secured communication by protecting request and response messages on CoAP level. This is in contrast to CoAP over DTLS that establishes secure channels between endpoints. OSCORE provides integrity, authenticity, and confidentiality by nesting the actual CoAP message as an authenticated and encrypted payload, interleaving information relevant to routing and retransmission in the unprotected outer parts. This layer hides sensitive information, such as the resource path and the CoAP method of the original message. Furthermore, the security of inner messages stays intact across protocol translations on gateways (*e.g.,* from CoAP to HTTP/S). OSCORE provides a strong request-response binding with mechanisms like sequence counters and sliding windows, which renders many attacks ineffective. The original specification is missing a key exchange protocol and thus does not provide perfect forward secrecy. Adaptations [249] allow for an Ephemeral Diffie-Hellman over COSE.

NDN authenticates response messages between arbitrary endpoints without the need for session state. While the application payload can be encrypted, NDN does not provide confidentiality for message headers. Moreover, NDN reduces security features to response messages only[4]. Names are an integral part of the NDN forwarding fabric and may contain sensitive application information. Thus, privacy concerns arise from plaintext names in NDN messages. An encryption or obfuscation of names inevitably affects the routing system and adds an exhaustive overhead. Unlike the CoAP variants, NDN follows the principle of immutable content: A specific name invariably points to the same content object. This property reduces the attack surface and desensitizes applications to delayed and replayed messages.

We summarize the observed advantages and drawbacks of the discussed protocol schemes

[4] Specification v0.3 is in progress and adds security features to Interests

Table 9.1: Summary of security properties for each protocol configuration. (✓) indicates optional specifications, which are unavailable in the used implementations.

	CoAP			**NDN**
	Protected	DTLS	OSCORE	Protected
Request Message				
Integrity	✘	✓	✓	(✓)
Authenticity	✘	✓	✓	(✓)
Confidentiality	✘	✓	✓	✘
Response Message				
Integrity	✘	✓	✓	✓
Authenticity	✘	✓	✓	✓
Confidentiality	✘	✓	✓	✘
Attack Resiliency				
Replay Insensitivity	✘	(✓)	✓	✓
Perfect Forward Secrecy	✘	(✓)	✘	✘

in Table 9.1, with a strong focus on the actual protocol behavior rather than on application payload security.

9.3.3 Message Overhead for Security

In all protocol configurations, security extensions add message overhead and consequently affect transmission times. Notably for IEEE 802.15.4, inflated messages easily increase media access times by a few milliseconds, whereas computational overhead in common IoT network stacks is in the range of microseconds [29]. We now quantify the overhead in terms of packet size which is introduced by the different security extensions. In Section 9.3.4, we will put this into perspective with respect to the common CoAP and NDN packets.

CoAP (Protected) and NDN (Protected) do not add any message overhead to requests. All configurations other than CoAP (Protected) add a structural overhead related to security. DTLS includes 11 bytes for the DTLS 1.2 record layer in all datagrams, excluding the epoch field. The NDN packet format uses flexible Type-Length-Value (TLV) fields to encode message headers. Security related TLVs similarly account for 11 bytes overhead. OSCORE exploits implicit information that results from a strong request-response binding and further utilizes a concise binary object representation (CBOR). This nets to a structural overhead of four and three bytes.

The security context identifier consists of two bytes for CoAP (Protected) and CoAP over DTLS. In the former scenario, contexts are identified by the 2-byte key identifier within our payload, while the latter scenario uses a 2-byte epoch field in the record layer to denote a secured session. OSCORE and NDN are able to reduce the length of small context identifiers. OSCORE

Table 9.2: Message overhead of security measures in bytes. Overhead does not apply to CoAP and NDN requests.

	CoAP						**NDN**	
	Protected		DTLS		OSCORE		Protected	
	Request	Response	Request	Response	Request	Response	Request	Response
Structure	–	0	11	11	4	3	–	11
Context ID	–	2	2	2	1	0	–	1
Nonce	–	2	8	8	1	0	–	0
MAC	–	8	8	8	8	8	–	40

omits the security context in response messages and requesting devices must deduce it from the request state.

For AES in CCM mode, the same nonce is required for encryption and decryption. The nonce is of variable length and usually ranges between 7–13 bytes [245]. We design our experiment to use partially implicit nonces [250]. Two bytes of the nonce are encoded into messages, while the remaining bytes are deduced implicitly, *e.g.,* from the hash of a resource URI. This allows 2^{16} messages per resource until a refresh of established security contexts is advisable. OSCORE repeatedly encodes smaller values in a single byte and CoAP (Protected) uses a 2-byte representation. In responses, OSCORE uses the same nonce to protect objects and thus omits the nonce. CoAP over DTLS uses eight bytes as a result of concatenating the epoch and sequence number fields. The remaining four bytes of the DTLS nonce are implicit and generated as part of the handshake process [247]. NDN benefits from the immutable content property: Since names always map to the respective content, its hash is used as nonce.

We use a message authentication code of eight bytes as defined by the TLS AES-CCM cipher suites [247]. NDN appends another 32-byte hash-based message authentication code (HMAC) signature that envelops the complete response packet. Table 9.2 summarizes the message overhead for the discussed protocols.

9.3.4 Security Overhead in Comparison to Basic CoAP and NDN Messages

We now dissect each message of the protocols under comparison in detail and relate the basic CoAP and NDN packet sizes to the security extension (see Figure 9.3). Our analysis distinguishes between requests and responses and includes all handshake messages for DTLS. We assume that a response payload includes a 2-byte temperature value.

IEEE 802.15.4 admits a maximum physical layer packet size of 127 octets. Assuming a typical configuration of 8-byte source and destination hardware addresses, considering a given 2-byte frame control field, 1-byte sequence number, 2-byte personal area network (PAN) identifier, and

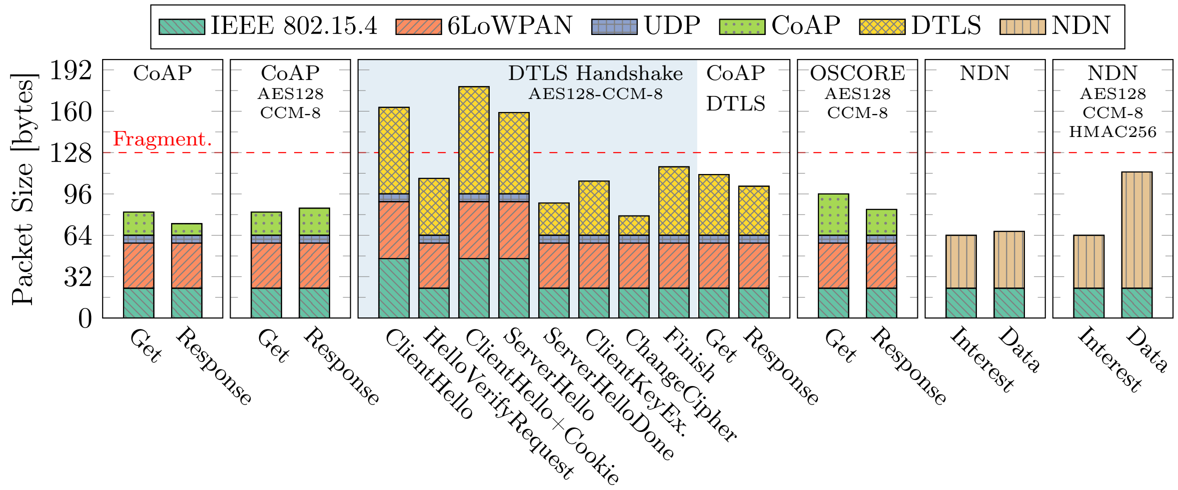


Figure 9.3: Packet structures of control- and data-plane packets for each protocol configuration.

a 2-byte frame check sequence, the total MAC header overhead adds up to 23 bytes for each protocol. This leaves 104 bytes for upper layer headers and user data.

In CoAP setups, the 6LoWPAN header occupies 35 bytes because it accommodates three 6LoWPAN dispatch bytes and two IPv6 addresses. Moreover, each packet counts six bytes for the compressed UDP header.

Special consideration is required for *ClientHello* and *ServerHello* packets in a DTLS handshake. In contrast to previous calculation, they surpass the maximum physical packet size and trigger a hop-wise 6LoWPAN fragmentation. While the MAC header overhead is therefore doubled, the 6LoWPAN overhead increases by only nine bytes for the inclusion of fragmentation dispatches in both fragments.

In contrast to unprotected CoAP responses, CoAP (Protected) messages inflate by 12 bytes to include the context id, nonce, and message authentication code of AES-CCM (see Section 9.3.3). CoAP over DTLS emits 29 and 27 more bytes for requests and responses, respectively, due to the DTLS record layer. OSCORE messages display similar but extenuating effects: requests increase by 14 and responses by only 11 bytes. The primary explanation for this surprisingly smaller increase is a reduced header overhead of OSCORE compared to the DTLS 1.2 record layer. Nonces are further omitted from responses to decrease their header overhead.

In contrast to CoAP, where responses exhibit smaller packet sizes than requests, NDN data packets exhibit larger sizes than Interests. This is a result of names being fully included in returning data packets. NDN data packets increase by an 8-byte AES-CCM MAC and an 32-byte HMAC signature, compared to unsecured NDN packets with an overall packet size of 64 bytes. Since Interest messages do not contain any security measures, their packet sizes remain unaffected.

9.4 Redundancy, Resilience, and Recovery

We now discuss prospective protocol features that are under early discussion in the IETF, or could be utile in the near future. The focus of this section is on protocol resilience against unanticipated service interrupts, fast and lightweight recovery, as well as on potential benefits from multi-destination content replication and caching.

9.4.1 Resilience to Loss of Security State

Nodes in the constrained IoT have frequent reasons to power down or even reboot. Protocol state which frequently updates such as session keys, nonces, or sequence numbers usually remains in main memory to reduce the number of energy-intensive I/O operations on non-volatile memory modules, and is therefore endangered to be lost. In distributed system settings such loss of state causes difficulties with synchronizing protocol behavior.

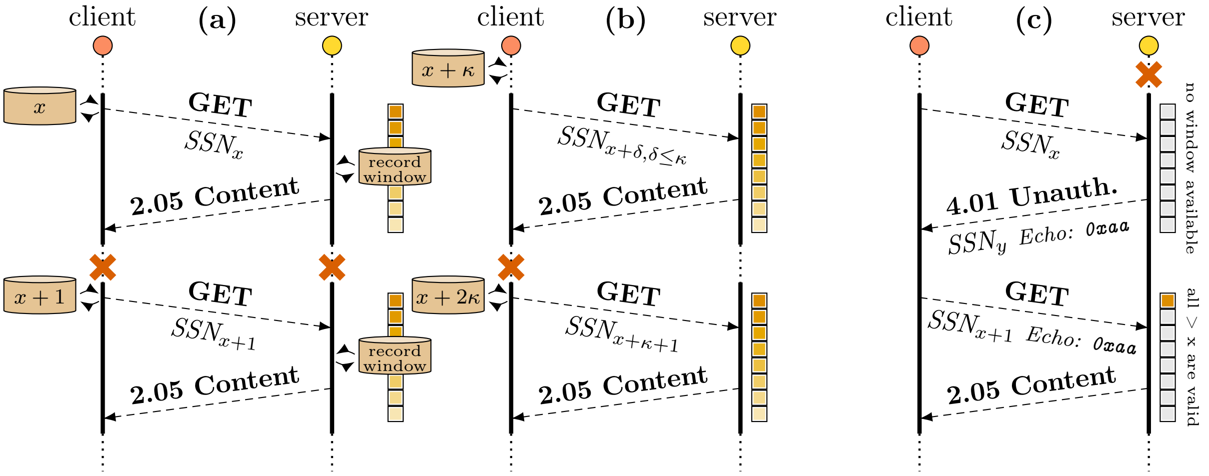
Security protocols usually manage state with varying demands on state persistence. Keying material is deployed statically at device bootstrap, dynamically exchanged, or derived from a key establishment scheme. These keys are considered immutable for the lifetime of a single security context and are persisted into non-volatile memory. Reorder and replay protection mechanisms require per-message state in the form of sequence counters, which update continuously with byte flow in a specific context. While these counters are essential for an efficient operation and remain effective across spurious system reboots, it strains energy and memory lifetime to preserve all individual state changes.

In this evaluation, we review the protocol behavior on unexpected security state loss for our selected protocol ensemble.

CoAP over DTLS deployments perform session establishment between endpoints to negotiate keying material and to agree on suitable cryptographic ciphers. Derived keys are valid for the lifetime of the session and are renegotiated on rare occasions, in which case the session epoch number gets incremented. The epoch number identifies delayed packets that did not properly transition to the new ciphers yet. Session keys and epoch fields easily persist on non-volatile storage to make them available throughout system reboots.

The sequence number in the DTLS record layer allows for detecting reordered and replayed messages below the CoAP layer. For this, each session endpoint individually advances a sliding window following its received sequence numbers. Since sequence numbers and the window bitmap are frequently updated, persisting them on a storage module is infeasible. A state loss due to unanticipated failure on either side thus requires a full handshake of six to ten message flights to re-establish the session. An optimization is the ticket-based session resumption feature [251], which reduces the handshake down to three flights. Tickets are created on the initial handshake and can be stored in non-volatile memory.

CoAP with OSCORE shares some DTLS basic properties, but has different recovery options. Keying material of a pre-established security context is infrequently updated and therefore



continuous persistence *sequence number recovery* *replay window initialization*

Figure 9.4: Protocol behavior due to loss of mutable OSCORE state after unanticipated client and server shutdowns with and without optimizations [26].

demands only a minimal amount of persistent storage I/O. Here as well, message sender sequence numbers (SSN) and sliding windows are employed to provide a replay protection mechanism. Only CoAP server nodes advance a sliding replay window for request messages.

Like in DTLS, an unexpected shutdown renders even persisted key material unusable when no recovery options are available. This hits OSCORE, however, harder than DTLS, for while DTLS can restart with a full handshake, OSCORE needs external mechanisms to arrive at a fresh security context. A simple but inefficient mechanism to avoid that is to persist the volatile state on each message, which adds to round trip times and memory wear. As shown in Figure 9.4 a, every single sequence number and each advancement of the replay window must be persisted to maintain a consistent state beyond an unanticipated system reset.

The appendix B.1 of the OSCORE specification [26] introduces two mechanisms to reduce the persistence operations to a few writes per reboot and takes the writes out of the request-response latency path, while introducing few or no randomness requirements. We distinguish between loss of volatile state on the client and the server side. On the client side (depicted in Figure 9.4 b), sequence numbers are leased in chunks of K numbers, and the last value of the leases is persisted. On system boot, or when the pool of leased numbers is near exhaustion, another chunk is leased and the persisted number increased. The interval provides a trade-off between costly write operations and the amount of sequence numbers that are lost after this procedure until a rekeying becomes necessary, and can be configured or adjusted automatically at runtime.

Figure 9.4 c depicts the behavior when a CoAP server loses the sliding replay window state. As the server does not persist a replay window, it cannot determine that any request is not a replay, and rejects it. Along with the rejection, it sends an encrypted fresh Echo value [241]. (While using a random nonce is an option, our implementation draws from the previously established sequence number pool). A client with the correct key material can repeat this request and mirrors the echo tag back to the server. On success, the server then accepts this request and initializes its replay window starting at a sequence number it knows to be fresh.

NDN IoT deployments typically use pre-shared keys or a complementary public key infrastructure to protect Interest and data messages. For deployments with asymmetric cryptography, content producers use key material to provide message authenticity and integrity by digitally signing content objects, while consumers require the corresponding public key counterparts to perform origin authentication on incoming data messages. A static configuration of necessary keys may work for a small set of devices in a network, but becomes unmanageable if content origins dynamically leave and join trust relationships. A trust schema [252] can be leveraged to configure automatic decisions for content producers and consumers to dynamically create, choose, and receive the correct key material based on the content names. In addition, temporary session keys can be derived from key exchange protocol extensions, such as OnboardICNg [253], or LASeR [254]. OnboardICNg builds on the authenticated key exchange protocol (AKEP2) [255] and LASeR is inspired by the extensible authentication protocol with session-key derivation using a pre-shared key (EAP-PSK) [256]. Both generate temporary security contexts that are valid for the lifetime of a session and the resulting keys can be used to maintain forward secrecy across consecutive sessions.

Similar to DTLS and OSCORE, persisting pre-shared secrets or session keys on device bootstrapping or session establishing allows for continuing a secured communication after unexpected system reboots. NDN does not include sequence numbers in messages, nor does it maintain frequently changing security state on devices, since the hop-wise content replication and immutability of named objects already reduce the attack vector for reordering and replay attacks (see Section 9.3.2).

9.4.2 Protected Multicast Device Discovery

Multicast on the network layer is a scalable mechanism for contacting large groups of possibly unknown devices. In fixed IPv6 networks it is an essential protocol feature for establishing communication relations (*e.g.,* using neighbor discovery [210]). In low-power, wireless regimes devices running 6LoWPAN face additional challenges.

LoWPAN nodes are often subject to disruptive events, *e.g.,* lossy links and device mobility, which make a perpetual connectivity between endpoints virtually impossible. Not uncommonly, reappearing nodes may configure new endpoint addresses due to expiring DHCP leases or privacy extensions [257] for stateless address auto-configuration schemes that limit the validity of addresses to a few hours or days depending on the scenario setup. A CoAP deployment thus usually requires a complementary infrastructure like the CoRE resource directory [9], which allows discovery of available resource endpoints. Nevertheless, in high mobility scenarios, propagation

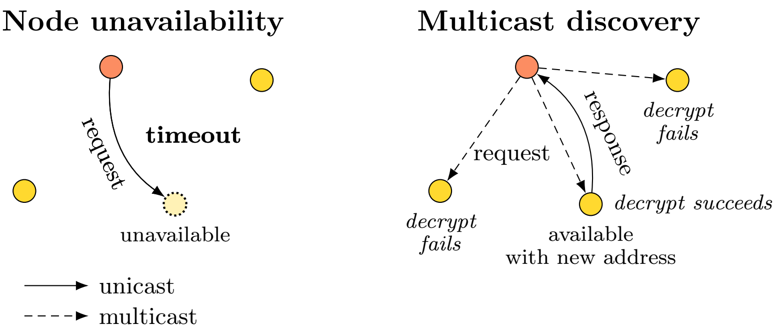


Figure 9.5: Node discovery with OSCORE using a scoped multicast request.

of frequently updating topologies to a central registry can introduce an infeasible communication overhead. A scoped multicast CoAP request is a lightweight alternative for (re-)discovering CoAP endpoints in close vicinity.

CoAP over DTLS prohibits the use of multicast addresses, because security sessions are bilaterally established between endpoint pairs. A multicast CoAP request is thus not protected by DTLS, which renders this lightweight alternative impractical for deployments with reasonable security demands.

OSCORE detaches the security context from endpoint-specific information. A request can be sent as a CoAP multicast message with OSCORE protection (in regular mode, without the Group OSCORE [244] extension) with its encryption and most privacy properties intact. Although such messages are potentially received by a group of nodes (see Figure 9.5), only a single server that holds the corresponding security context can decrypt and respond. The response possibly returns via unicast to the client node. On a successful transaction, subsequent requests can use this newly discovered unicast address.

To stay reachable after an address change, an OSCORE server could inform its peers of an identifier that is usable for longer than its network addresses. This may happen by explicit announcement, or as additional information when the OSCORE context is set up. As such an identifier can be used to track a device across address changes and possibly across different OSCORE keys, its privacy implications need to be considered before employing such a scheme.

Carefully designing group memberships in wireless deployments with multicast support is pivotal to reduce energy expenditure and excessive media utilization. Following the IPv6 address architecture [258] (see Figure 9.6), a 16-byte multicast address decomposes into a 2-byte address classifier and a 14-byte group identifier. The `ff02` classifier identifies multicast addresses that are link-local.

We propose a scheme that maps OSCORE state into the least significant bytes of an IPv6 multicast address and corresponding nodes configure these addresses on their network interfaces. This optimization utilizes scoped multicast messages that are already filtered on the network layer and only recipients with likely-to-match OSCORE contexts will receive and process them.

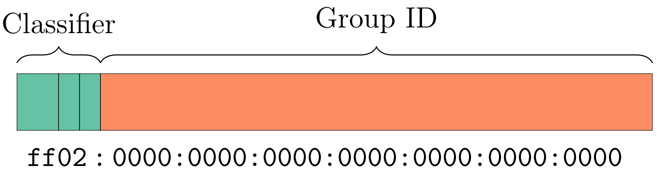


Figure 9.6: IPv6 link-local multicast address format [258] using a 2-byte classifier and a 14-byte group id.

We propose link-local identifiers that map to one or more OSCORE contexts. These identifiers could be derived from the OSCORE key ids, or could be distributed via an external mechanism, *e.g.,* during a session establishment process, or could utilize global node identifiers, if available.

Choosing a suitable length within the multicast address is a trade-off between two extremes: With too short identifiers, collisions may force many nodes to receive and try to decrypt messages. Long identifiers possibly consume too much of the available address space. To find an advisable id length, we follow the design decision for the `solicited-node` multicast address [258], which is used in the neighbor discovery protocol of IPv6 [210]. The address has the prefix ff02::1:ff00:0/104 and allows for a 3-byte (24-bit) variability in the low-order.

We use the same amount of variability by using three bytes long identifiers, but configure another multicast prefix to not interfere with the neighbor discovery. An example prefix could be the currently unallocated ff02::3:ff00:0/104 address (see Figure 9.7), but any serious deployment would need to check with an up-to-date IANA registration to sustain interoperability with coexisting protocols.

```
ff02:0000:0000:0000:0000:0003:FFXX:XXXX
```

Figure 9.7: Link-local multicast addresses for OSCORE context discovery.

The base of 2^{24} addresses makes it unlikely that a device needs to process an OSCORE request intended for another device. More precisely, the probability of collisions is known from the birthday paradox. Equation 9.1 describes this probability for a year of d ‘days’ and n ‘people’. In our case, $d = 2^{24}$ is the given address space and n is the number of multicast addresses in use. Evaluating equation 9.1 yields a collision probability $< 1\%$ for 575 multicast addresses, which must be considered a large number of security groups on a single link.

Still, in case of a collision, the full key id that is part of any request serves as a further distinction. Eventually, the authenticated encryption of OSCORE filters out remaining collisions.

$$p(n; d) = 1 - \prod_{k=1}^{n-1} \frac{d-k}{d} \tag{9.1}$$

NDN allows for secured multicast messages similarly to OSCORE, provided a proper name to MAC address mapping [109] is in place. In contrast to OSCORE, a multicast request discovers content and not devices. Since content is matched by exact and immutable names, no additional mechanisms to protect against re-ordering or replay are needed, even though a single request can lead to returning responses from several content producers or in-network caches. NDN deduplicates multiple data messages of the same request on the forwarding plane by serving only the first incoming data and thereafter discarding all replicated messages.

9.4.3 Protected Multi-Source and Multi-Destination Messages

Scalable group communication is essential for supporting key IoT use cases: Multi-source readings of uniform sensors are most efficiently implemented as a convergecast following a multi-destination read request. Groups of actuators are often coordinated via multi-destination control messages. Over the air (OTA) firmware updates are vital for device maintenance and hard to implement without efficient multicast flows.

Multicast communication is an inherent property of the information-centric NDN architecture. Endpoint information is absent from the addressing scheme, which allows for direct multi-source and multi-destination access by applications mapping to a multicast scheme. In NDN, a multi-source communication is enabled as follows: nodes on intersecting request paths aggregate requests and returning responses fan out on the interfaces to the corresponding requesters. Multi-destination requests, on the other hand, are supported by on-path caches and multiple fan-outs in the forwarding information base for the same name prefix.

Responses that traverse a request path consume the request state. Thus, responses are deduplicated on path intersections when arriving from multiple destinations, and only the first response is forwarded. Since requests and responses are protected on an object level, the security measures of NDN are equally effective in unicast and multicast communication.

The classic CoAP request-response model enables multi-destination requests by leveraging IP multicast groups for requests with response messages returning via unicast. In contrast to NDN, a reliable multicast communication using acknowledgments and retransmissions is unsupported, since the number of endpoints within a multicast group is unknown in IP multicast. CoAP proxies cache requests from multiple clients for the same resource, and aggregate observations by fanning out single responses to multiple clients using unicast messages. Sending a request to a multicast address does not preclude caching, but practical deployment via a proxy depends on experimental extensions [259] and the client awareness of the origin being a multicast location. Setups where the client is unaware of that and a reverse proxy requests from multicast to return the first response are conceivable, but we are not aware of any existing implementation or experimentation.

With OSCORE, multicast requests are covered by Group OSCORE [244], necessitating a client awareness of the multicast context. Work on distributing responses to multiple clients is highly experimental [260, 261]. The approaches are centered around the clients obtaining or building identical requests to which the responses can be bound. Such request-response mappings are necessary because knowing the request is essential for understanding the response. They promise to transfer both the caching and aggregation abilities of CoAP over to OSCORE, and even to extend the base CoAP mechanism to allow multicast distribution of responses. We further evaluate the impact of group communication with caching in our experiment report in Section 9.5.5.

9.5 Evaluation in the Testbed

In this section, we compare the different protocol configurations based on real implementations deployed in a testbed.

9.5.1 Experiment Setup

Scenarios & Parameters. We want to quantify the performances of a protected CoAP and NDN communication in a typical IoT data collection scenario with multiple sensor nodes. For this, subsequent requests periodically traverse a gateway into an IoT stub network. Each sensor device is requested 1000 times at an (randomly jittered) interval of $2 \pm 0.5s$ and returns a 2-byte temperature reading. To allow for comparison of pull-based NDN with CoAP, we limit CoAP methods to confirmable GET.

We align our experiments with respect to retransmission and timeout configurations. All protocols employ the same retransmission strategy: On failures, nodes wait two seconds before retransmitting the original request. In NDN, retransmissions are performed hop-by-hop, while CoAP performs them end-to-end. At most four retransmissions will occur for each data, which is the default configuration for CoAP [4, Section 4.8]. Remaining protocols do not have standard defaults and we align with CoAP.

We do not consider congestion from external cross-traffic in this work. However, each individual transmission experiences self-induced background traffic from on-going requests and retransmissions. The jittered request interval further mixes the event space and allows a greater exploration of the state space. On average, this cross-traffic is constant per experimental run.

Software & Hardware Platform. All devices run RIOT version 2019.10. NDN deployments are based on CCN-lite, and CoAP experiments use the default GNRC network stack of RIOT including libOSCORE and tinyDTLS (*c.f.,* Section 9.2).

We conduct all experiments on the FIT IoT-LAB [27] testbed. The hardware platform consists of class 2 devices [1] featuring an ARM Cortex-M3 MCU with 64 kB of RAM and 512 kB of ROM. Each device is equipped with an Atmel AT86RF231 [69] transceiver to operate on the IEEE 802.15.4 radio. The testbed provides access to several sites with varying properties. We perform our experiments on the *grenoble* site in a single-hop and multi-hop configuration. Our single-hop setup consists of one gateway node and ten sensor nodes in broadcast range as illustrated in Figure 9.8. In the multi-hop configuration, we use one gateway node, ten sensor

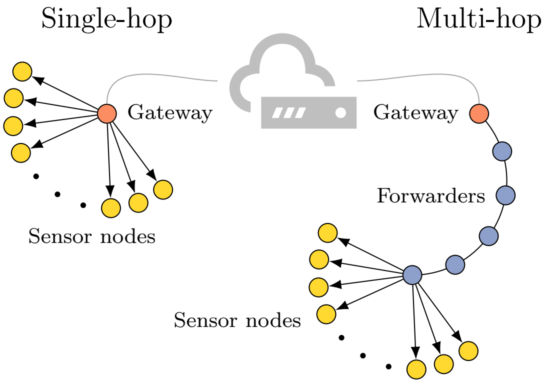


Figure 9.8: Topologies for single-hop and multi-hop experiments.

nodes, and five forwarder nodes. Forwarding states are statically configured on each node to form the topology depicted in Figure 9.8.

Protocol Configurations & Start-Up Conditions. In all setups, we use AES in CCM mode with a 128-bit key and limit the resulting message authentication code to eight bytes as described in [247]. Each configuration also contains a 1-byte key identifier where applicable. The NDN (protected) setup further includes a hash-based message authentication code (HMAC) salted with a pre-shared key. We limit the number of security contexts on the gateway to ten and on each sensor node to one. As a consequence, sensor devices maintain only one DTLS session concurrently and all secured content objects from a particular sensor device use a single security context. As we do not evaluate key management schemes, we compare all security protocols with pre-shared keys. Context related variables such as sequence counters are set to default on device start-up.

9.5.2 Time to Content Delivery

We examine the delays measured between content requests and content arrivals at the gateway. Figure 9.9 combines the results for CoAP and NDN configurations in the single-hop and multi-hop setup. We first observe that the protocol families are in rough accordance for the single-hop case. Temporal performances indicate a sub-second completion time for close to 100% of all transmissions across the protocols. The unprotected NDN configuration displays the fastest operation with 50% of transmissions finishing below 11 ms. Combining this observation with our previous result that NDN transmits the smallest request and response messages (see Figure 9.3), we can conclude that unprotected NDN succeeds in quickly exchanging its small messages. In the unprotected CoAP configuration, 50% of transmissions finish below 13 ms. The protected protocol versions follow closely, whereby CoAP over DTLS is on the slow end with 50% of transmissions finishing only below 16 ms.

Next, we consider the more challenging multi-hop scenarios. Overall, results reveal much

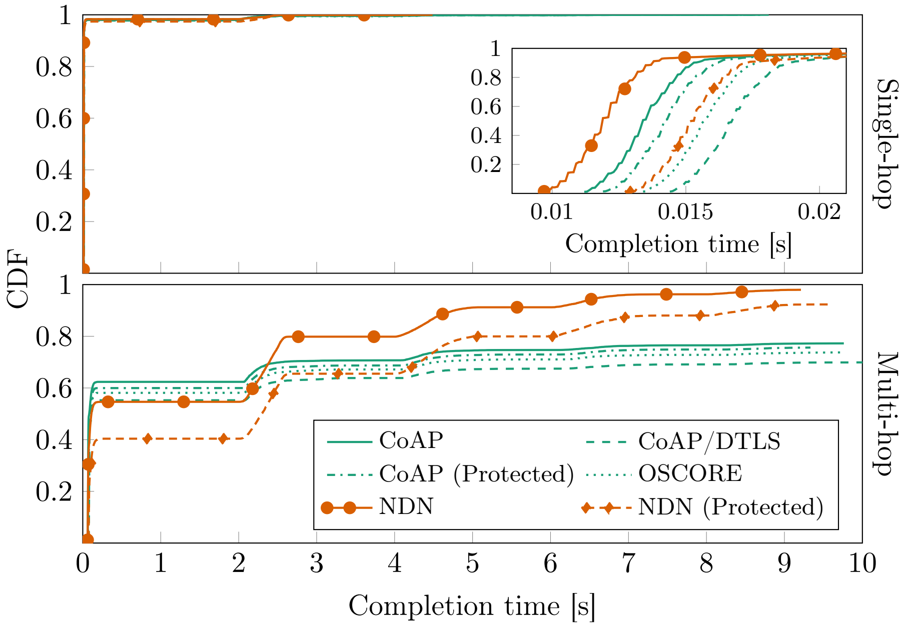


Figure 9.9: Temporal distributions of content arrival times.

slower protocol operations. This reflects the common experience in low-power regimes that radio interferences and individual error probabilities accumulate over several hops and decrease reliability for the entire subnet. The staircase pattern visible for all protocols is based on request retransmissions at the configured interval of 2 s per retransmission. On the slower end, stairs show attenuating effects due to an accumulating jitter for each retransmission. We observe that all CoAP variants operate in agreement. Roughly 55–60% of content requests complete in the sub-second range without requiring retransmissions. Corrective actions, *i.e.,* request retransmissions, delay the completion time, but are able to increase the number of successful responses at the gateway to 70–77%.

The effects of inflated messages also become apparent. CoAP keeps packets smallest and reveals a better performance than CoAP over DTLS, which requires the largest packets. The delay distributions for NDN show surprising results. Only 40–55% of all responses arrive in the sub-second range at the gateway and therefore seem to indicate an inferior performance of NDN compared with the CoAP variants. This discrepancy is nevertheless due to the different retransmissions strategies of CoAP and NDN. NDN implements hop-wise retransmission, which stepwise increases the packet numbers on the forwarding path and makes interference with parallel content requests on the wireless links more likely. Hop-wise retransmission with in-network caching, however, advances packets toward receivers in each step and converges more easily to successful content delivery. Hence, corrective actions are able to boost the overall reliability of the NDN family to 92–97%.

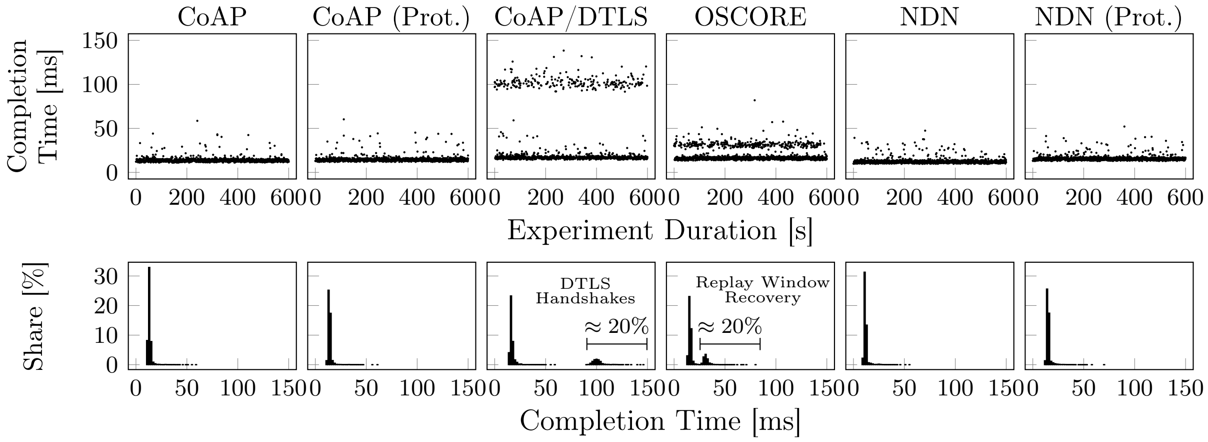


Figure 9.10: Content arrival times and their percental distribution during the evolution of an experiment in a single-hop scenario, with simulated reboots after 20% of all exchanges. DTLS and OSCORE clearly reflect these events as delayed handshakes or reply window recovery, respectively.

9.5.3 Security Overhead

We now inspect the case in which an endpoint repeatedly connects to the IoT stub network to retrieve sensor readings. This setup follows the previous single-hop scenario with one minor change: The endpoint at the gateway simulates an unexpected reboot by losing any volatile cryptographic state after 20% of all exchanges. (With OSCORE, the simulated unexpected reboot happens at the sensor node instead, as an unexpected reboot of the gateway would have no visible effect at all). Figure 9.10 summarizes the evolution of content arrival times throughout an experiment duration of ten minutes. In the top row, we measure times to completion during the ongoing experiments for all protocols, while the bottom row visualizes the distributions of the completion times as measured for each protocol. The supplementary Table 9.3 delivers an in-depth view on the statistical key properties of the arrival time distributions in Figure 9.10, including the total average μ, standard deviation σ, the first quartile Q1 (25%), third quartile Q3 (75%), and the median.

Mostly the completion times reflect the results already presented in Figure 9.9. Neither our protected or unprotected CoAP deployments, nor the NDN counterparts use volatile security state. The distributions of all transmission times of the experiment thus accumulates at around ≈12–15 ms, even for the case of unexpected reboots.

An obvious exception to this is CoAP over DTLS. After a loss of cryptographic state at the gateway, a session handshake must precede the initial request. Such handshake for the configured cipher suite requires ten DTLS packet transmissions in total. Figure 9.3 depicts the composition of these packets and clearly shows their considerable sizes. In some cases, handshake messages are much larger than the actual authenticated and encrypted user packets.

Protocol	μ [ms]	σ [ms]	Q1 [ms]	median [ms]	Q3 [ms]
CoAP	13.93	2.83	12.82	13.48	14.43
CoAP (Protected)	14.72	2.94	13.57	14.22	15.18
CoAP/DTLS	34.09	34.00	16.29	17.27	23.07
OSCORE	16.42	2.72	15.23	16.08	17.00
NDN	12.44	2.91	11.28	11.95	12.89
NDN (Protected)	15.79	3.15	14.52	15.33	16.15

Table 9.3: Statistical key properties of content arrival times in milliseconds for successful requests in a single-hop scenario with simulated reboots.

Our evaluation shows that DTLS handshakes complete after around 100 ms, whereas in rare cases they even require up to 150 ms. Since these handshakes are measured on a single hop, they clearly serve as a good lower bound for DTLS negotiations in more complex scenarios.

A proper DTLS session resumption [251] and Connection IDs [239] could reduce the effects of handshakes after deep sleep or address changes, but are not available in tinyDTLS and require the persistence of context information that scales with the number of connected sensor devices.

The OSCORE recovery process also displays a delay introduced by the state loss. The effect, though, is much less pronounced as state recovery completes within a single round-trip as depicted in Figure 9.4 c, and does not establish a fresh security context. The messages involved are not depicted in Figure 9.3 for lack of a visible difference to the regular OSCORE messages, as their sizes differ only by up to 4 bytes throughout the tests.

9.5.4 Request Creation Time

We conclude our protocol measurements by evaluating the message creation time of requests. We start our measurement when an application triggers a request and stop the time as soon as the packet is passed to the lower layer, which is UDP in the CoAP variants and the link-layer in NDN. Figure 9.11 visualizes the results using two bar plots for each protocol. The first bar shows average creation times for initial requests and the second bar denotes average creation times for request retransmissions.

CoAP in its unprotected and protected configurations exhibits the lowest creation times at around 280 μs. Since the protected version only affects responses, equal times for requests are expected. In both protocol versions, retransmissions are built much quicker in around 43 μs, since they already exist in retransmission buffers. NDN behaves similarly, but creation times increase to $\approx$ 810 μs for initial requests and $\approx$ 95 μs for request retransmissions. The latter is

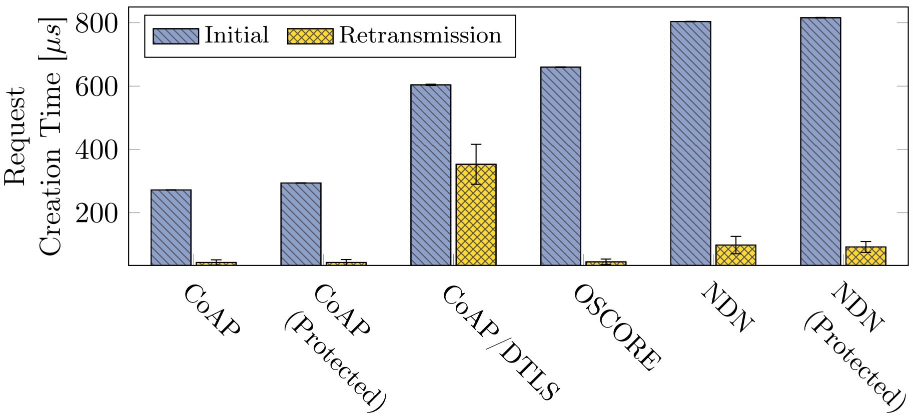


Figure 9.11: Creation time for initial and retransmitted requests.

mainly due to complex (TLV) header structures, which require string parsing, and significantly less optimized implementations of packet processing in CCN Lite.

The behavior of CoAP over DTLS differs. Initial request creation times escalate to around 600 μs due to the authenticated encryption. In particular, request retransmissions reveal outlying results. Requests exist in CoAP retransmission buffers and reduce overall creation times, but they still require to pass through the DTLS layer. Hence, retransmission creation times spend on average around 353 μs and therefore eight times as long as other CoAP setups. This problem arises from layering independent protocols, which is not present in the OSCORE. Protection takes place on the CoAP layer and retransmission buffers already contain protected messages. Creation times for retransmissions reside at around 45 μs and are thus comparable to the protected and unprotected CoAP composition.

9.5.5 Impact of On-Path Caching

In our previous evaluation of temporal performance (see Section 9.5.2) we could observe that hop-wise (re-)transmission with content caching of NDN yielded higher success rates than the host-centric protocols in this low-power and lossy wireless regime. Notably, CoAP already brings the architectural feature of CoAP proxies with caches, which can be used to construct a deployment that is hop-centric like NDN.

Content object security integrates seamlessly into hop-by-hop deployments with untrusted CoAP proxies, whereas transport layer security requires proxies to be included in trust relationships to utilize response caches. We envision OSCORE a natural candidate and a potential enabler for protected CoAP deployments that align with information-centric concepts such as on-path caching. This can increase the robustness of CoAP in networks with intermittent connectivity.

In a preview of future prospects and developments, we examine the effectiveness of content replication on request paths. For demonstration purposes, we consider an extended topology

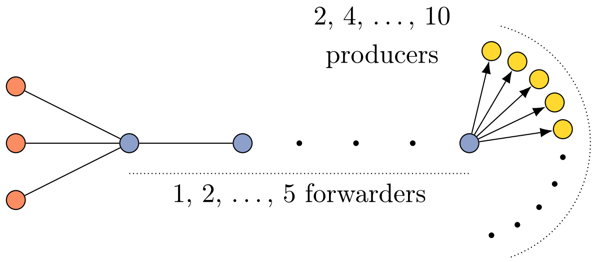


Figure 9.12: Topology with three consumers and a varying number of forwarders as well as producers to gauge the effects of hop-wise caching.

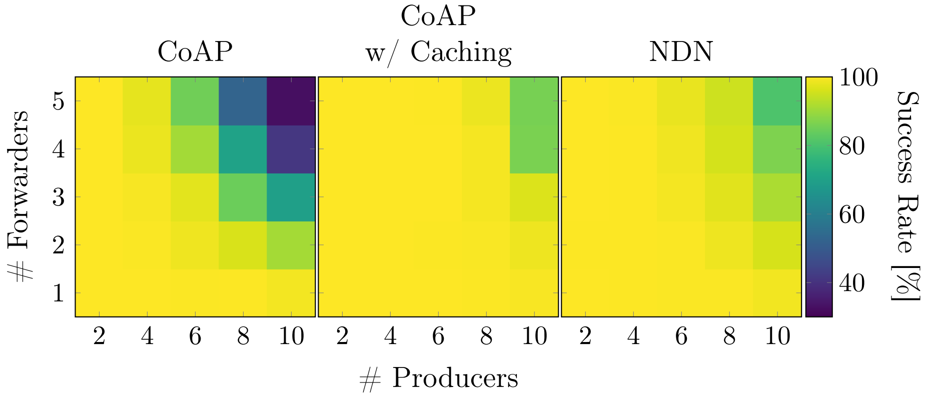


Figure 9.13: Success rates for plain CoAP compared to alternative deployments with hop-wise caching capabilities.

(Figure 9.12) that consists of three consumer nodes, a number of forwarders varying from one to five, and a number of content producers increasing from two to ten. All producers host ten different temperature readings and all three consumers request every 2–3 seconds a random content item out of these ten sensor readings for a period of 100 requests per producer. Content caches for the extended CoAP deployment and for NDN are dimensioned to hold 30 content objects.

We compare CoAP with and without caching and NDN in Figure 9.13. Success rates of data delivery are averaged over the three consumers. In all three deployments, we record success rates of close to 100% for simpler topologies. For the more complex topologies starting at three forwarders and six producers, we observe success rates that quickly collapse down to ≈ 40% for the plain CoAP deployment, while the by caches extended CoAP and NDN setups are able to remain operationally successful at rates of ≈ 80% and above. The increased robustness in the presence of on-path caching has two reasons. First, caching shortens request paths for retransmissions of the same request, and second, content is pulled closer to the consumers, such that subsequent requests to the same CoAP resource do not need to fully traverse the request path. The small fluctuations in the success rates between NDN and CoAP with caching for the setups with more than six forwarders is due to different qualities of their implementations and their varying behavior under increased network stress. Furthermore, the open testbed infrastructure is shared by multiple users and interferences are unpredictable and not suppressible.

These phenomena have been known in the information-centric network world for years [111]. Content object security enabled by OSCORE may introduce these network optimizations soon into the CoAP ecosystem.

9.6 Conclusions

In this chapter, we explored the vision of content object security on the network protocol level for the IoT. We analyzed current protocols and prospective developments that aim for securing content independent of network transport, and measured different configurations of CoAP, OSCORE, and NDN in a real-world testbed of constrained nodes. With this comprehensive study, we spanned the full solution space from end-to-end security sessions that act on transport channels to approaches that secure each data chunk individually. Our findings indicate that in challenging environments those protocols that can hop-wise transfer and cache content objects significantly outperform protocols which cannot. Smoothly integrated into the CoAP mechanics, OSCORE identified itself as a promising candidate for the former.

We further identified security overheads and the burdens from volatile and non-volatile protocol state that warrants session persistence. As explicit replay and reorder protection in NDN are unnecessary, energy-expensive I/O operations to non-volatile storage can be minimized and caching simplifies. In contrast, DTLS and OSCORE require up-to-date sequence numbers and sliding replay windows to prevail even after unexpected system resets, which marks these protocol elements as candidates for future optimization.

As we show both the impact of overheads and of hop-by-hop retransmissions, the present results help to justify, guide and (in future work) evaluate further improvements in the compared protocols: The upcoming DTLS 1.3 will optimize the record layer footprint [262]. OSCORE extensions that allow for (re-)establishing a security context (LAKE [242], EDHOC [263]) will need to keep extra round-trips to a minimum. Cacheable group observations [260] enables NDN-like multicast features in CoAP, which would be beneficial as discussed in this work.

Chapter 10

Data Orientation in CoAP Deployments

Abstract

The information-centric networking (ICN) paradigm offers replication of autonomously verifiable content throughout a network, in which content is bound to names instead of hosts. This has proven beneficial in particular for the constrained IoT. Several approaches, the most prominent of which being Named Data Networking, propose access to named content directly on the network layer. Independently, the IETF CoAP protocol group started to develop mechanisms that support autonomous content processing and in-network storage.

In this chapter, we explore the emerging CoAP protocol building blocks and how they contribute to an information-centric network architecture for a data-oriented RESTful Web of Things. We discuss design options and measure characteristic performances of different network configurations, which deploy CoAP proxies and OSCORE content object security, and compare with NDN. Our findings indicate an almost continuous design space ranging from plain CoAP at the one end to NDN on the other. On both ends—ICN and CoAP—we identify protocol features and aspects whose mutual transfer potentially improves design and operation of the other.

10.1 The Problem of Building an Information-Centric IoT and Related Work

The Internet of Things (IoT) increasingly connects embedded controllers built into intelligent machines and at the same time drives a huge deployment of sensor devices, which collect and report measurements from the wild. This massive machine-to-machine communication exchanges syntactically and semantically well-structured data for further aggregation and processing in some cloud. This data-centric nature at the Internet edge called for rethinking the current IoT architecture [99], and emphasized consideration of information-centric principles in the future IoT development.

Coping with Constraints. The mass constituents of the IoT will be tiny, cheap *things* that communicate via low power and often lossy channels. The IETF has designed a suite of protocols that adapt to this constrained environment. The IPv6 adaptation layer 6LoWPAN [3] enables a deployment on constrained links (*e.g.,* IEEE 802.15.4), which RPL routing arranges in a multi-hop topology [202]. The Constrained Application Protocol (CoAP) [4] offers a lightweight alternative to HTTP while running over UDP, or DTLS [25] for session security. This set of solutions extends the host-centric end-to-end paradigm of the Internet to the embedded world and puts IPv6 in place for loosely linking the *things.*

ICN networks have been early identified as a lean and efficient network alternative for a future IoT [110, 199, 98]. Popular operating systems for low end IoT devices such as Contiki [107] and RIOT [157] have been providing NDN network stacks [198, 108] for years. ICNLoWPAN [264], an adaptation layer for NDN and CCNx for constrained wireless links, has been designed and outperforms 6LoWPAN. Hence from a resource perspective, the information-centric concepts and software solutions have well met the challenges posed by the low end IoT edge.

Adapting Communication. Many IoT access networks are wireless, slow, and error-prone. In this context, the original end-to-end design of the Internet [63], which pushes service functions such as reliability up to the transport, turns into a challenge: Several retransmissions via multiple hops quickly exhaust network resources and interfere with subsequent communication requests.

Name-based routing, hop-by-hop forwarding, and in-network caching have been shown to support robustness of application scenarios in regimes of low reliability and reduced infrastructure (*e.g.,* without DNS). In comprehensive experiments, network caches established as efficient retransmission buffers, which significantly decreased network load and improved the overall network performance [111]. Several cache optimization strategies for an information-centric IoT [265, 152] could improve the overall network performance and resilience even further.

Securing Content Objects. Adding security credentials to content objects instead of transmission channels is a new approach to secure communication on the Internet. Information Centric Networking first introduced content object security on the network layer for the sake of ubiquitous caching. Recently, the IETF Core working group released OSCORE, which extends the IoT ecosystem to content object security.

OSCORE [26] is a protocol extension to CoAP and addresses the issue of security terminating at gateways. Instead of securing sessions between endpoints, OSCORE protects entire CoAP messages and provides integrity, authenticity, and confidentiality on an object level. The original CoAP message is thereby encapsulated as an authenticated and encrypted COSE [240] object by an outer CoAP option. OSCORE utilizes the request-response semantics of its underlying CoAP layer and an elaborate nonce construction to obtain compact response messages. Recently it was shown that OSCORE message protection clearly outperforms DTLS session security in the constrained IoT and approximates the NDN performance in several dimensions [66].

Shaping a Mainstream Technology. Many forces drive the current development of the IoT and lead to a rather fragmented protocol landscape. Historic domain-specific (local) protocols, traditional industry standards, and the present IETF suite all persist in specific deployments. The traditional request-response content access is the popular approach for the current IoT [82]. It is foreseeable that soon a standard solution will be desired to ensure interoperability between the steadily emerging new applications and deployments. With this in mind, Fotiou *et al.* [266] developed a CoAP emulation that runs over ICN. Keeping ICN as the underlying network preserves beneficial concepts and technologies that have been developed over the past dozen of years.

The alternative approach is to transfer the insights, design elements, and features established in ICN research to the current IETF standards and transform the protocol composition and its deployment into an ICN variant. We will show in the following that the CoAP protocol suite is almost ready to host an information-centric web of things while complying to the well established Internet standards.

10.2 CoAP versus ICN: A Feature Set Comparison

10.2.1 Security

Request-Response Binding. A key aspect of ICN is its ability to address content objects instead of traditional network endpoints. In the most prominent ICN expressions, CCNx and NDN, names bind immutably to content. This immutability allows for a range of positive effects, including a long liveliness with regard to caching purposes, a resource-friendly content provenance validation using digests [267], and a desensitization of applications to delayed and replayed messages. Since content requests are considered idempotent, a transactional request-response binding is not required.

CoAP follows the RESTful model and is architecturally akin to HTTP. Requests contain URIs that resolve to service endpoints, which can serve static or dynamic content. Request methods add further semantics to requests and allow for state transitions in the application. Non-cryptographic tokens in the CoAP header match responses to corresponding requests. With security mechanisms layered below CoAP (*e.g.,* the widely deployed datagram transport layer security DTLS [25]), applications need to actively manage their tokens to fend off attacks. Otherwise, the inability to provide a verifiable request-response mapping can be fatal, especially in cases where resources publish mutable content [246]. The OSCORE security layer establishes verifiable message binding, and the upcoming Request-Tag option [241] extends it to fragmented request representations.

Object-level Provenance and Encryption. The integral caching component of ICN systems enables content retrieval from potentially untrusted peers. On that account, most ICN solutions implement data integrity, provenance, and origin authentication on the protocol level [126]. Access control, authorization, and privacy on the other hand are challenged by this pluralistic networking approach and are left to upper layers or the application. In CCNx and NDN systems, security measures are generally applied to returning response messages. Both architectures also allow for the inclusion of digital signatures in request messages.

CoAP by itself does not include any security measures, but was designed like HTTPS to rely on transport layer security by (D)TLS. As a protocol extension, OSCORE protects entire CoAP messages and provides integrity, authenticity, and confidentiality on an object level. The original CoAP message is thereby encapsulated as an authenticated and encrypted compressed COSE [240] object.

Résumé. *ICN authenticates content independent of its consumers, whereas CoAP OSCORE binds security to individual access requests by authenticating and encrypting CoAP messages.*

10.2.2 In-Network Caching

Cache Model. The immutability of content objects and a name-based routing as applied by CCNx and NDN allow for a seamless integration of on-path content caching in the network. While a ubiquitous caching with adequate cache replacement strategies reduces access times of popular content, it offers one additional benefit that is strikingly valuable especially in lossy environments: caches serve as retransmission buffers in order to boost the content delivery reliability. Retransmissions generally happen on the scale of seconds, *i.e.,* allocated cache space is short-lived and quickly released.

CoAP proxy endpoints [4, Section 5.7] can store messages on two conceptually separate layers[1], in message deduplication and in an application layer cache. Each networked device along a path can operate as a proxy, which will generate a cache distribution similar to ICN.

Messages secured by OSCORE are strictly bound to a single request. Hence, they can only be meaningfully retained in CoAP proxies for message retransmissions. Proxies are not allowed to see details of content as required to find suitable cache entries from previous transmissions. Clients—even the same client served by an older response—lack the context to decrypt it. Efforts to adapt OSCORE group communication to produce cacheable requests are underway, but have not yet produced testable results.

Content Freshness Model. CoAP uses a freshness model that is comparable to the content freshness handling of CCNx and NDN. A CoAP Max-Age option in responses provides a lifetime hint for caching endpoints, after which this response is marked as stale.

Content Validation Model. CoAP applies an efficient validation model to revalidate stale responses using the ETag [4, Section 5.10.6] option in request messages. Instead of transmitting the full response, a validating origin server merely responds with a small message to indicate whether a cached response is considered to be valid again. In contrast, CCNx and NDN have

[1] The unified design of CoAP as a single protocol spanning both cache layers allows caching at one layer to be foregone in many cases.

no notion of invalid cache entries, since named content is immutable and can only expire, but not change.

Résumé. *ICN binds names to immutable content for long-term, in-network caching, whereas CoAP proxies cache on a message level, including optimized signaling for validation.*

10.2.3 Request Handling and Forwarding

Message Synchronization. The ICN design decision to address content independent of its location complicates the temporal decoupling of request and response messages. In name-based routing architectures, the requesting and requested endpoints are unknown. Responses travel along a reverse path that is temporarily constructed from the request. Long intervals between request and response require equally long-lived soft-states on each hop in the network. NFN [268] and RICE [269] are two protocol extensions that support a handling of long-running requests, but long-lived Interests place a burden onto the network.

Plain CoAP deployed between endpoints requires state only at these endpoints. Conversely to CCNx and NDN, the reception of requests is acknowledged by the content producer. Such open requests at the consumer can easily be long-lived and allows the producer to respond proactively as soon as content is available.

Reliable Transport. Both protocol families support retransmissions following message timeouts initiated by the requester. For the ICN protocols, retransmissions are not bound to endpoints but happen from hop to hop. If previous requests have populated the on-path caches, retransmissions benefit from cache hits, which pull the content closer to the requester.

Following the host-centric paradigm, CoAP uses end-to-end retransmissions, but can deploy caching proxy nodes to enhance reliability of the transport. On-path caches rebuild a hop-wise content replication and thereby benefits of ICN.

ICN and CoAP support similar features to report on error cases. CoAP encodes error codes into response messages analogous to HTTP. CCNx specifies an Interest Return message and NDN delegates the error reporting to NDNLP [115].

Next-hop Selection. ICN designs typically use content names to perform next-hop lookups in a Forwarding Information Base (FIB). In the common end-to-end CoAP deployment, requests are forwarded based on a destination IP address matched against a FIB.

When CoAP is used with proxies, forwarding decisions are performed on the application level. RESTful Web protocols have established mechanisms to include forward proxies in a network autoconfiguration using WPAD (Web Proxy Auto-Discovery Protocol) [270] and to decide the next-hop based on the host name of the resource using the PAC (Proxy Auto-Config) feature, which is implemented in all common web browsers. No such mechanism has yet been described for the IoT, but the application of analogous techniques seems plausible. Such a mechanism could in particular be used to learn the next-hop from the underlying discovery protocol as a forward proxy, if it was discovered that the capability is available there.

The forwarding decision is usually based on the authority component of the request URI. That typically, but not necessarily contains a resolvable host name. Nodes that cannot resolve an authority component (*e.g.,* because they do not implement DNS) often rely on a default proxy that handles name resolution for them.

DoS Protection. A central design aspect of NDN was to prevent the submission of unwanted content, which has the beneficial effect of making traditional Denial of Service (DoS) impossible [21]. For this, one important building block is the absence of endpoint addresses, which makes it harder to target packets to a specific node. In a dense deployment of CoAP proxies (*i.e.,* a proxy on each forwarding node) very similar techniques can apply. For the next-hop proxy, nodes only need to resolve its link-local address from the FIB, which in turn will be elided by the 6LoWPAN header compression – hence leaving the packet without network address.

Unfortunately, it was soon discovered that stateful Interest forwarding in NDN can lead to a different kind of DoS attack [191, 127], which was later coined 'Interest Flooding' [271]. CoAP proxies are susceptible of similar attacks that inflate state and overutilize CPU and memory resources.

Résumé. *Both ICN and CoAP with dense proxy deployment can perform a request routing on names (URIs) and a stateful content forwarding. A cache-assisted reliable transport option is available for both families.*

10.2.4 Multi-Source & Multi-Destination

Multicast. Support for multicast communication is an inherent property of most ICN implementations. The absence of endpoints in the addressing scheme allow for multi-source and multi-destination classes of applications with virtually no added overhead. In popular ICN systems, multi-source communication is designed by aggregating requests at nodes on intersecting request paths. Returning responses fan out to the corresponding requesters. Multi-destination requests are supported due to on-path caches and multiple target entries for the same name prefix in the forwarding information base. Responses that return from multiple destinations are dropped at path intersections as soon as existing request states are consumed by the first response.

CoAP supports group communication using IP multicast [272] as the underlying data transmission [243, 273]. In contrast to the more nuanced multicast integration of ICN designs, CoAP disallows confirmable multicast messages. Retransmissions in case of message timeouts are thus delegated to the application. Current research [259] looks into leveraging proxy nodes to support fan-outs of unicast messages at the proxy and takes up on the open question of how to handle multiple returning responses. One approach is to leave the deduplication of multiple returning messages to the application, while another approach is to aggregate multiple responses at the proxy to return a common message to the requester. This technique would not only be applicable to routable multicast addresses, but also to proxies that have multiple forward routes for a given resource and authority URI component, allowing setups analogous to ICN architectures with multiple destinations for a prefix.

Mobility. Multicast mobility is an asymmetric problem [237]. While the movement of receivers is often easy to accommodate by local network reconfigurations, the impact of mobile sources on the routing is complex and requires assisting measures or services. In a network setup with proxy nodes, multicast proxy services have proven useful in orchestrating network reconfiguration [274, 275], as they adapt locally with only link-local signaling on the control plane. It is expected that these techniques can be transferred to CoAP proxies in a straight-forward manner. Mobility in ICN [164, 166] sees the analogous problem space. It is easily supported for consumers and difficult to implement for content providers.

Protected Group Messages. Object security as commonly implemented in ICN-based systems integrates with the intrinsic multicast support and allows for a seamless group communication with secured messages. The responsibility for a proper key management is entrusted to the deployment.

CoAP is commonly deployed with DTLS in order to provide secure communication channels between endpoints. The end-to-end nature of DTLS complicates a group communication by design. OSCORE brings object security, but the strong binding between the request and response excludes a multicast operation. Ongoing research [244] extends the OSCORE model to tolerate source authentication for CoAP group requests and the corresponding responses.

Résumé. *ICN and (unprotected) CoAP support multicast communication, whereas multi-party communication for proxy-assisted CoAP is still in its design phase.*

10.3 Deployment Scenarios

We deploy NDN and different compositions of CoAP protocols as schematized in Figure 10.1. Starting with plain CoAP `GET` requests, we gradually add more and more protocol features of ICN-nature to approach the NDN setup. Protocol operations and configurations are detailed in the following.

10.3.1 Standard NDN

Hop-by-Hop Request. Common NDN deployment uses a name-based routing, hop-wise requests, and on-path caches.

Hop-by-Hop Retransmission. Each hop on a request path arms a retransmission timer for Interests. If content is not timely returned, then the initial Interest is repeated. NDN integrates message deduplication and request aggregation features in order to suppress the transmission of Interests for request paths that are already set up.

Object-Level Security. Security on an object level is inherent to NDN. While the outer response packet can be signed using different cryptographic algorithms, an HMAC signature

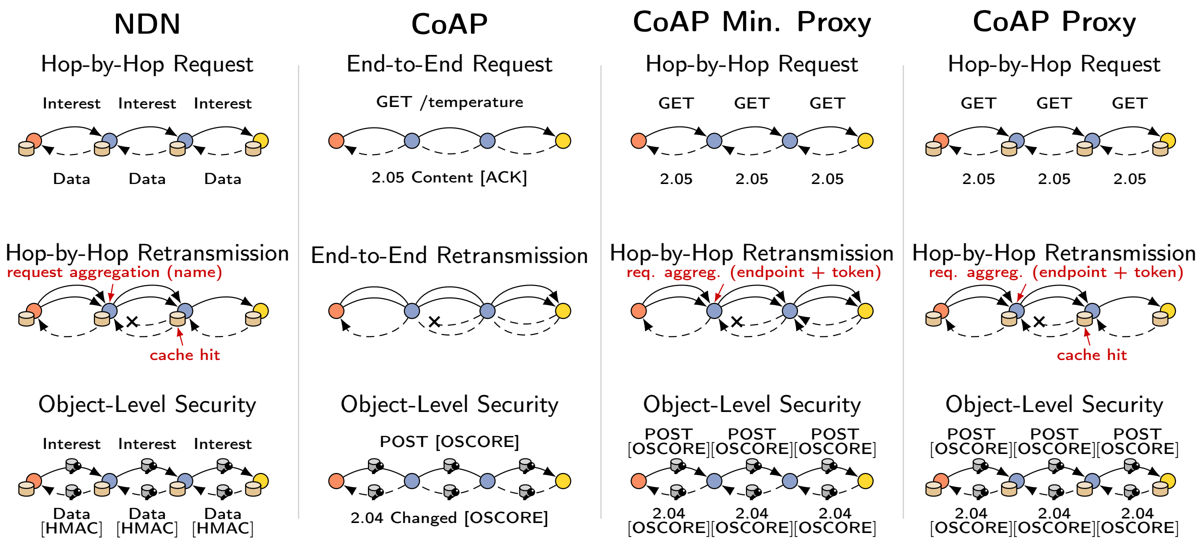

NDN
Hop-by-Hop Request
Interest Interest Interest
Data Data Data
Hop-by-Hop Retransmission
request aggregation (name)
cache hit
Object-Level Security
Interest Interest Interest
Data [HMAC] Data [HMAC] Data [HMAC]
CoAP
End-to-End Request
GET /temperature
2.05 Content [ACK]
End-to-End Retransmission
Object-Level Security
POST [OSCORE]
2.04 Changed [OSCORE]
CoAP Min. Proxy
Hop-by-Hop Request
GET GET GET
2.05 2.05 2.05
Hop-by-Hop Retransmission
req. aggreg. (endpoint + token)
Object-Level Security
POST [OSCORE] POST [OSCORE] POST [OSCORE]
2.04 [OSCORE] 2.04 [OSCORE] 2.04 [OSCORE]
CoAP Proxy
Hop-by-Hop Request
GET GET GET
2.05 2.05 2.05
Hop-by-Hop Retransmission
req. aggreg. (endpoint + token)
cache hit
Object-Level Security
POST [OSCORE] POST [OSCORE] POST [OSCORE]
2.04 [OSCORE] 2.04 [OSCORE] 2.04 [OSCORE]

Figure 10.1: Deployment scenarios and protocol configurations used in our comparative evaluations.

seems most appropriate for the IoT. Encrypting the content within responses is left to the application.

10.3.2 Routed CoAP

End-to-End Request. CoAP supports different request methods, from which `GET` compares best to Interest requests of NDN. Unlike in NDN, request state exists only at the endpoints.

End-to-End Retransmission. `GET` requests can be issued unreliably (`NON`) and with corrective actions enabled (`CON`). In `GET CON`, each request requires an acknowledgment, which is piggybacked in the response message. On absence, a retransmission of the initial request message is triggered.

Object-Level Security. OSCORE provides a secure communication between two endpoints. `GET` requests are nested into COSE objects and are cryptographically secured. These objects are then included in CoAP `POST` messages as OSCORE objects. The returning response is treated similarly by nesting the message into a COSE object and delivering the OSCORE object in a `2.04 Changed` response.

10.3.3 CoAP with Minimal Proxy

Hop-by-Hop Request. CoAP proxy nodes operate at the application level and handle conversions between CoAP and other protocols. A proxy runs as a `reverse`, or a `forward` proxy and is commonly situated at the network edge. Requests that traverse a proxy intermediately terminate and lose their end-to-end semantics between endpoints. Responses follow the same request path through the proxy node in reverse—a property which is well-known from ICN approaches, such as NDN.

In this scenario, we install forward proxies on all forwarder nodes. The minimal version in this scenario is included for illustrative purposes, and lacks message deduplication and storage as is regularly required with CoAP.

CoAP clients include `Proxy-URI` options in request messages to provide forwarding hints to the proxies. This option contains the URI string that encodes the URI scheme, the authority component that identifies the CoAP server, and the service path. Each proxy manages forwarding state and passes requests either to subsequent proxies, or to the origin server. In case the request arrives at the final proxy node, the message is translated for normal CoAP operation, *i.e.,* the Proxy-URI string is split into its URI components and a common `GET` request is transmitted to the origin server.

Hop-by-Hop Retransmission. The CoAP specification does not fully outline the proxy operation for request retransmissions, but we envision the following two scenarios: First, a proxy acknowledges the reception of the request using an empty acknowledgment message and thus pauses any further retransmissions of the previous hop. Second, a proxy identifies incoming retransmissions based on the token and endpoint information. It then aggregates duplicate requests to the outstanding request state. Concurrently, the proxy handles its own retransmissions. For our deployment setup, we consider the latter approach as it approximates the NDN operation quite well.

Object-Level Security. Messages secured with OSCORE require no additional interaction on proxy nodes. As with various other options, the OSCORE option is copied to the reissued request and thus forwarded until it reaches the designated endpoint. The requested service path name resides within the encrypted security envelope of OSCORE and is not accessible from the outer CoAP message. This does not only protect the authenticity of request-response exchanges from attempts of tampering and forgery, but also retains privacy by hiding the requested path from eaves-droppers. To maintain these security properties, Proxy-URI strings outside the security envelope only contain the URI scheme and authority sections, but not the service path. In secured OSCORE deployments, CoAP proxies thus make forwarding decisions based on less information than in unsecured deployments.

10.3.4 CoAP with Proxy

Hop-by-Hop Request. The addition of retransmission caches to each forward proxy on a path advances the protocol transformation: This deployment shows huge similarities with NDN in terms of hop-wise message passing and hop-wise caching. A CoAP message deduplication module aggregates requests based on the message correlation parameters. From those, it determines whether a response is already being processed, and does not forward it.

Hop-by-Hop Retransmission. In our setup, no separate responses [4, Section 5.2.2] are

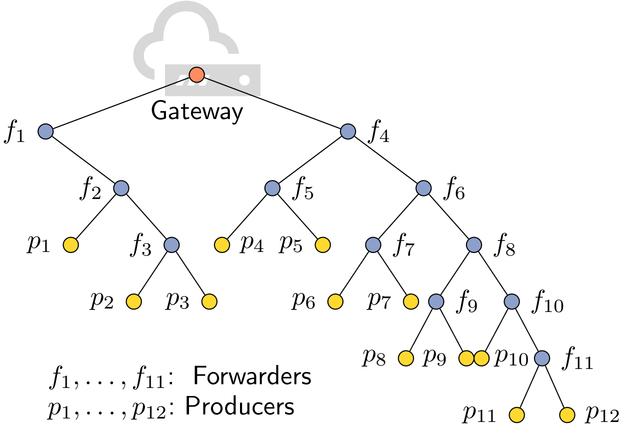


Figure 10.2: Topological arrangement of gateway, forwarders, and producer nodes for each deployment.

used. Thus, the responsibility for ensuring that the response arrives stays with the client. The response content is cached in the proxy at least as long as request retransmissions by the client are expected.

Object-Level Security. OSCORE messages are encapsulated in CoAP `POST` requests and `2.04 Changed` responses. Those are stored in the retransmission cache.

10.4 Evaluation in the Testbed

In this section, we quantitatively assess the five deployment scenarios outlined in Section 10.3 using real protocol implementations and experiments in a testbed.

10.4.1 Experiment Setup

Use Case and Topology. Our experiments follow a typical IoT application: A consumer node is situated at the network edge (the gateway) and retrieves sensory data (*e.g.,* temperature readings) from content producers. A set of forwarder nodes provides connectivity between the consumer and producers. The gateway, forwarder, and producer nodes statically arrange on system startup in a Destination Oriented Directed Acyclic Graph (DODAG) as illustrated in Figure 10.2 for all protocol deployments. DODAGs are optimized for the predominant converge cast scenario, *i.e.,* they yield shortest paths from sensors to cloud services, but show suboptimal paths for sensor to sensor traffic. Our setup has a total of 12 producers and 11 forwarder nodes. This minimal constellation can already show signs of link and memory exhaustion on network stress. In this topology, the caches closer to leaves experience less load, while caches near the gateway show cache replacements much more frequently.

Deployment Parameters. In our experiments, the gateway periodically issues requests via

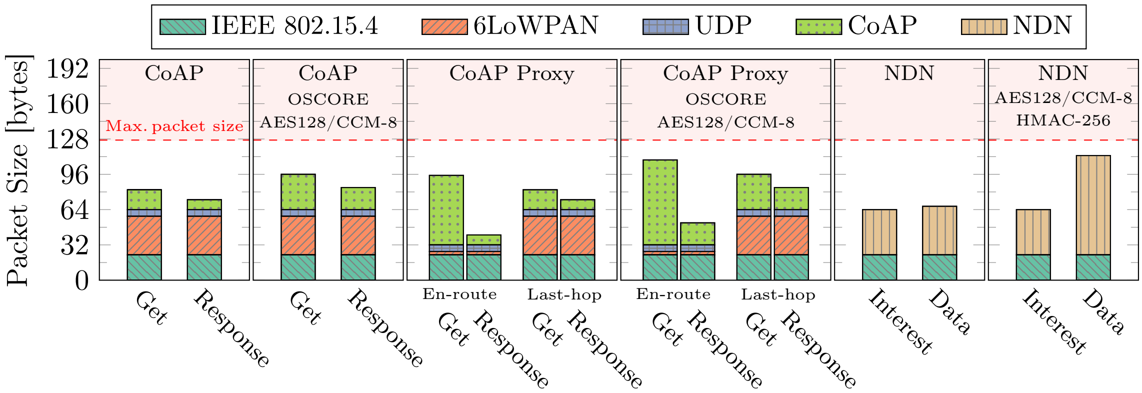


Figure 10.3: Packet structures and sizes of data-plane packets for each protocol configuration.

the IoT stub network to its edge sensors. Each sensor device is requested 500 times at an interval of 1.25s ± 0.1s and returns a 2-byte temperature value. That time was chosen such as to create a situation of pronounced network load for all scenarios. All experiments are aligned with respect to retransmission and timeout configurations. On message timeout, nodes wait two seconds before initiating a retransmission of the initial request; retransmissions are limited to five. In this work, we do not add explicit interferences from external cross-traffic. Still, each individual transmission experiences background traffic from ongoing requests and retransmissions that are self-induced by the experiment. Requests are jittered, though, to mix the event space and to allow for a better state exploration.

Software & Hardware Platform. All devices run RIOT [28] version 2020.04. NDN deployments are based on CCN-lite and CoAP experiments use the default GNRC network stack of RIOT including libOSCORE[2]. The CoAP forward proxy is an additional software module and was extended for caching.

We conduct our evaluations on the FIT IoT-LAB [27] testbed. The testbed hardware consists of class 2 devices [1] featuring an ARM Cortex-M3 MCU with 64 kB of RAM and 512 kB of ROM. To operate on the IEEE 802.15.4 radio, each device is equipped with an Atmel AT86RF231 [69] transceiver.

10.4.2 Message Overhead

We first dissect the details of request and response messages of the examined protocols in Figure 10.3. We fix the response payload to a 2-byte temperature value.

The maximum physical layer packet size of IEEE 802.15.4 is 127 bytes. In our interface configuration, the total MAC header overhead adds up to 23 bytes, leaving 104 bytes payload size for upper layer protocols.

In all CoAP deployments, the 6LoWPAN overhead accounts for 35 bytes, which carry the

[2] `https://gitlab.com/oscore/liboscore`

dispatch types and two global IPv6 addresses. A single exception are packets forwarded between CoAP proxies: they can use link-local IPv6 addresses as discussed in Section 10.2.3, which 6LoWPAN can elide by header compression. In this case, the 6LoWPAN overhead reduces to the remaining three dispatch bytes. The compressed UDP header requires an additional 6 bytes.

Request messages in the standard CoAP deployment require 18 bytes for the application layer, which includes the resource URI string `/temperature` and CoAP related protocol information, such as the 2-byte message ID and the 2-byte token. In contrast, response messages display the much smaller packet size of 9 bytes. This is a result of omitting resource URIs in the response and use the 2-byte token to match returning responses to corresponding requests.

Content object security with OSCORE deployment inflates request messages by 14 bytes and response messages by 11 bytes due to security encoding overhead and a message authentication code. An OSCORE protocol optimization allows the same nonce values for cryptographic operations on requests and responses. With this, the nonce value is completely omitted from response messages, as they are obtained from the request state on the requesting node.

The forward proxy deployment of CoAP uses the `Proxy-Uri` option string in requests to designate an endpoint. In contrast to the plain CoAP, sizes of the CoAP protocol increase by 45 bytes for CoAP requests in the unsecured and secured cases. The last forwarder hop prior to the producer node transforms the Proxy-URI string into appropriate CoAP options. Since at the same time messages on the last hop use a global IPv6 destination address, 6LoWPAN needs to include the additional 32 bytes for the addresses again. Response messages do not include any forwarding hints and compare to response sizes of the regular CoAP deployment.

NDN keeps requests smallest with a protocol overhead of 41 bytes. This includes the name. Due to its design that mirrors request names back in Data messages, responses tend to exceed the packet sizes of their requests. The secured variants inflate the message sizes by 46 bytes for Data messages, which is significantly more expensive than OSCORE. Interest messages are not affected by security measures and do not include a security overhead.

10.4.3 Time to Content Arrival

We measure the times to complete a content request, *i.e.,* the time from requesting content to its arrival at the gateway. Note that this metric summarizes not only the speed of protocol data transmission, but also the distribution of loss events and the effectiveness of corrective protocol actions. Figure 10.4 displays the corresponding distributions for our compared protocol deployments with and without content object security in place.

We first observe that all protocol families are in rough agreement with the configured retransmission intervals. Distributions in the sub-second range represent transmissions that succeed within one round-trip. Retransmissions operate in a two-second interval and lead to the staircase pattern observed for all protocols, but CoAP NON. The unreliable CoAP NON protocol is able to successfully complete more requests ($\approx 75\%$) than any other protocol on the first try, which

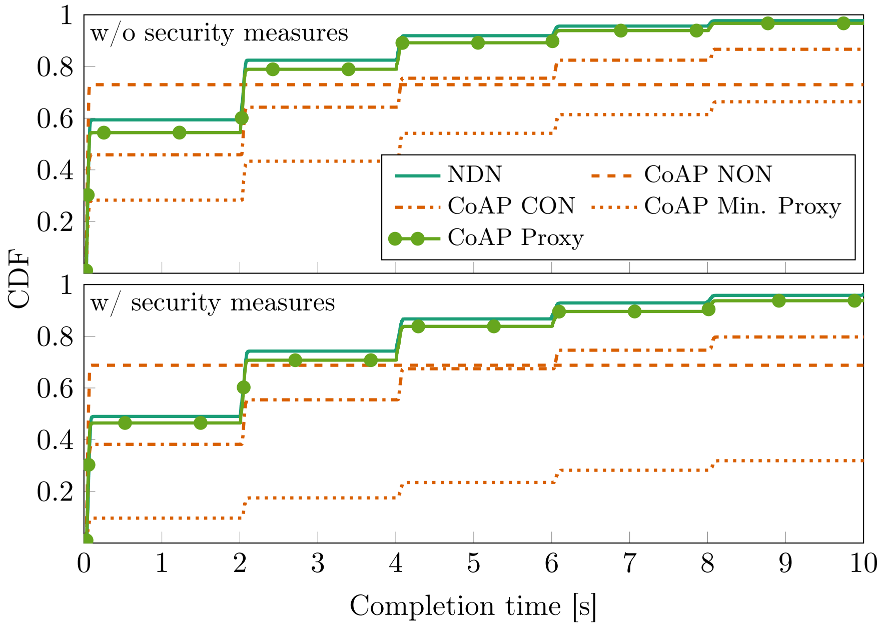


Figure 10.4: Time to content arrival—distributions for different protocol deployments and security settings.

in turn is due to its unreliability: The lack of retransmission control keeps the medium free of retransmissions, hence leaving more capacity to the initial packet transfer. Content security overhead reduces the success of CoAP NON to ≈ 66%.

The reliable protocols CoAP CON and hop-wise CoAP minimal proxy similarly fail in completing the sensor readings within five retransmissions. CoAP minimal proxy operations yield a rather poor temporal distribution with final success rates of 70% (w/o security) and 30% (w/ security). In this setup, the increased packet sizes, but foremost the hop-wise retransmission requests amplify the link stress immensely to a point, where no reliable communication between producers and the gateway is possible. In contrast, CoAP CON shows higher success rates than CoAP minimal proxy due to a lower retransmission control overhead: End-to-end retransmissions sequentially traverse all hops of a path until they reach a destination, or a packet loss occurs. With hop-wise retransmissions, messages originate independently of the previous hop, as long as forwarding state exists from previous attempts.

In contrast, the full CoAP proxy deployment exhibits a success rate of 98 % and performs very similarly to NDN. The secured versions show temporal performances that match the distributions of the unsecured cases. Due to the increased message sizes, success rates decrease minimally for all protocols, except for the hop-wise CoAP minimal proxy operation. It is clearly visible that the full CoAP proxy can leverage the potentials of hop-by-hop transfer with intermediate retransmissions served by the caches just as NDN does.

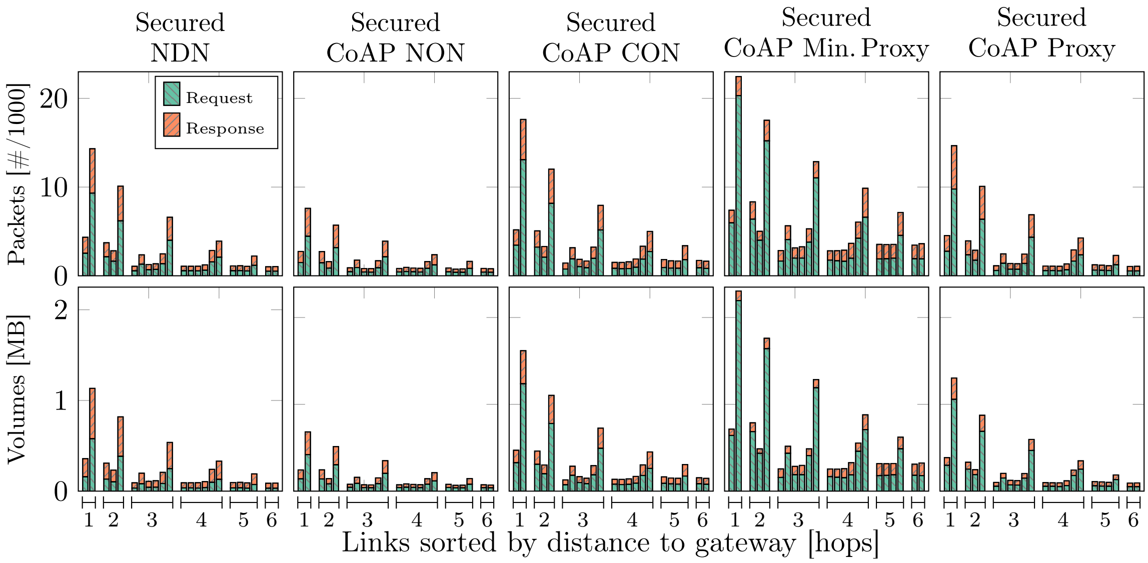


Figure 10.5: Number of packets and bytes transmitted over the air per downstream and upstream link in the topology for each deployment with security measures enabled.

10.4.4 Link Stress

The topology in Figure 10.2 generates different levels of link stress for regular communication throughout the network. We measure packet events and total bytes over the air for each protocol and link in the topology using independent sniffer devices. Note that our links are within an overlapping broadcast domain with mutual interferences. Since our experiments use carrier sensing of the radios within a static topology, we argue that our measurements of captured unicast traffic between device pairs serve as a proper estimate on the protocol induced link stress.

Figure 10.5 displays the results for the secured protocol variants. All links are grouped by their hop distance to the gateway node and we further distinguish between request (link downstream) and response (link upstream) packets. At first, different values in each link group are due to the number of nodes in the sub-tree served by each link.

CoAP NON displays the fewest number of packets and even lower data volumes on each link, which is expected due to its lack of retransmission capabilities and smaller packets. All other protocol scenarios show slightly more request packet events than responses. Hop-wise CoAP minimal proxy in particular generates a much larger number of request messages than responses. This is due to many request retransmissions triggered by intermediate proxies and corresponds to our observations in the completion time measurements (see Figure 10.4).

NDN and the full CoAP Proxy show similar results of captured packet events per link and a similar relation between requests and responses. Data volumes, however, differ noticeably: Posing a request is much cheaper in NDN than in CoAP due to the packet structure given in

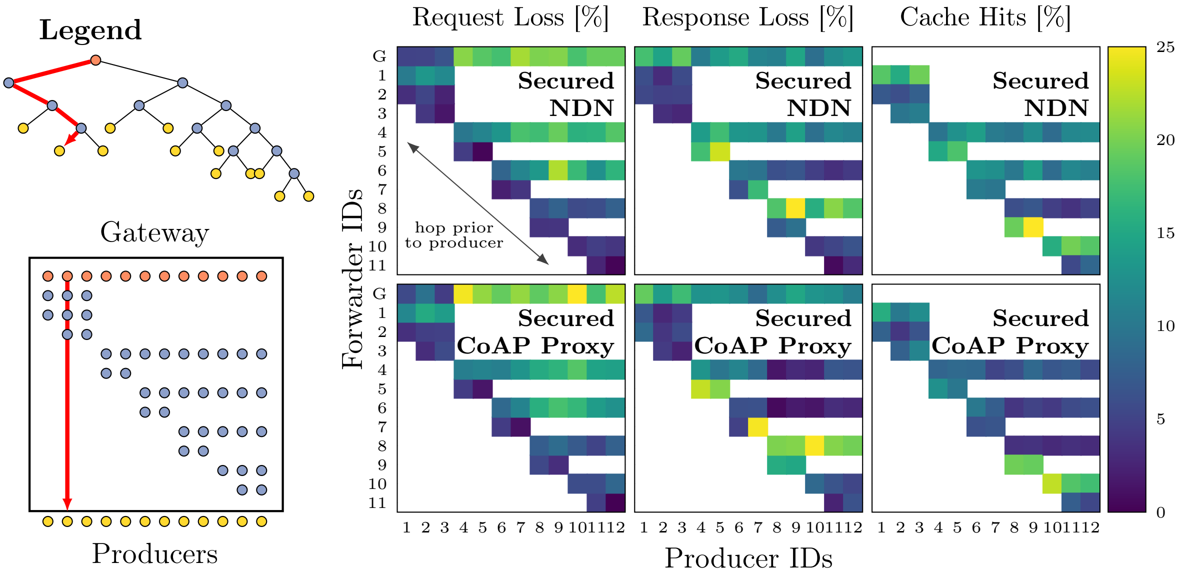


Figure 10.6: Packet loss and cache hit ratios for secured NDN and CoAP Proxy. Columns represent request paths from top (gateway) to bottom (producer) as illustrated in Figure 10.2.

Figure 10.3. This uneven link utilization is the result of (*i*) unsecured Interests, which keeps requests small for NDN, and (*ii*) the additional 32-byte HMAC and the message authentication code for the NDN payload, whereas OSCORE displays a much smaller security footprint.

10.4.5 Cache Utility

We now confront cache utilization with packet loss on each hop for the secured NDN and full CoAP Proxy. These metrics disclose how effectively transmission failures can be compensated by a nearby cache. Results are displayed in Figure 10.6. Each cell in the matrix describes the message exchange between a node and its downstream neighbor toward a particular producer. A column from top to bottom represents a valid request path from the gateway to a producer across a varying number of forwarders as illustrated in Figure 10.2.

We first observe the request losses per link, which cannot be compensated from caches. The overall picture reveals that NDN better succeeds in delivering requests to the next hop, which is expected due to the smaller request message sizes (see Figure 10.3). CoAP clearly shows to be at a disadvantage with higher efforts in delivering requests, reaching relative loss rates up to 20–25%.

On the message response side, the converse holds: NDN performs slightly worse compared to CoAP, which again can be attributed to the increased message sizes of NDN Data. Looking at the cache hits, we are interested in how efficient caches compensate for data loss. Ideally, a

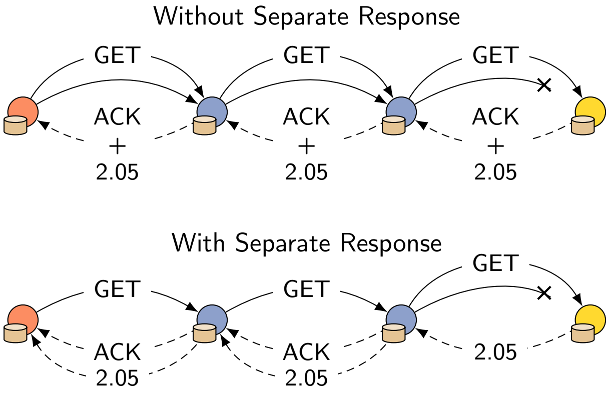


Figure 10.7: The mechanism of early acknowledgments as separate response to reduce hop-wise request retransmissions.

lost response can be served from the next-hop cache on the path, *i.e.,* a bright cell in the loss matrix is shadowed by an equal brightness at the next populated cell of lower Y-coordinate.

Caching services nicely work for NDN: Response losses on the line $Y = 4$ for example are recovered by the caches on line $Y = 6$ (the next populated), and the high loss at coordinate (9,8) is immediately serviced from the next cache (9,9). Cache services are less pronounced for CoAP, since data losses are less pronounced. Also by accident, one lossy link (7,7) directly connects to a producer without intermediate cache. On the overall CoAP shows a fair cache utility, as well.

10.4.6 Early Acks in Separate Responses

Confirmable CoAP messages are retransmitted until an acknowledgment arrives, or a message timeout occurs. For confirmable requests, CoAP allows to piggyback acknowledgments in returning data responses. This is the preferred mode if data responses are generated immediately. Separate response [4, Section 5.2.2] is a protocol enhancement in CoAP to pause unnecessary request retransmissions of the client in case the response generation takes longer than the configured request message timeout. In this mode, an empty acknowledgment message is promptly sent to the requesting client. Once content is available, a response with the actual content is then returned.

In this evaluation, we want to quantify the control overhead of the secured CoAP proxy deployment and compare it to a deployment variant that uses separate responses as illustrated in Figure 10.7: each CoAP proxy is configured to immediately acknowledge an incoming GET request, while the origin server responds with a piggybacked acknowledgment as before.

Figure 10.8 shows the frequency of outgoing requests—including request retransmissions—that originate from the gateway node and incoming responses received by the gateway over the duration of the experiment. Our first observation is that CoAP without an early acknowl-

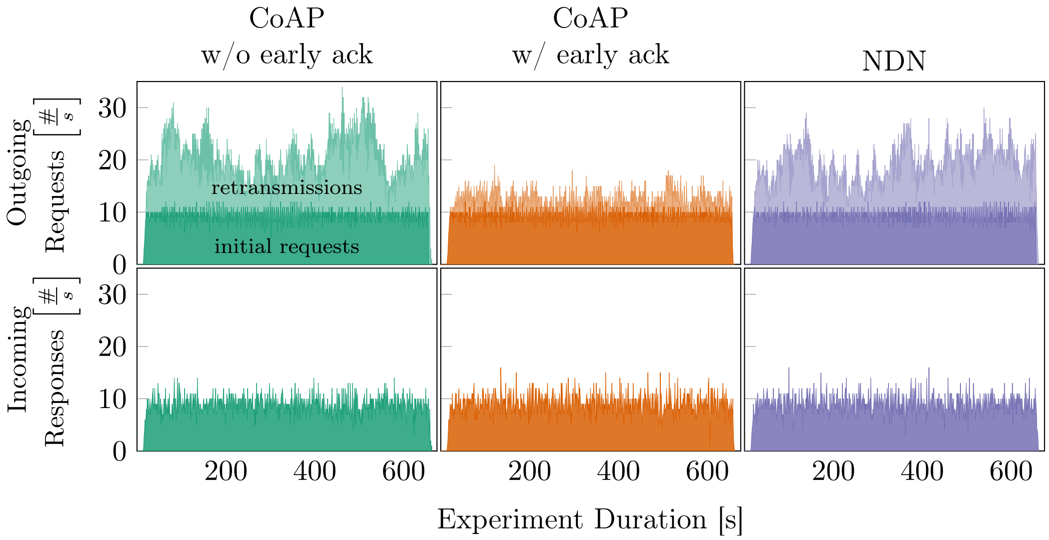


Figure 10.8: The effect of early acknowledgments in CoAP—numbers of outgoing requests and incoming responses measured at the gateway node.

edgment mechanism and NDN display similar performance. This is consistent with our results in Figure 10.4 and Figure 10.5. Both deployments employ the same hop-wise retransmission strategy and show analogous completion times as well as success rates. It is expected that the request to retransmission ratio is also comparable.

In detail, we observe a number of outgoing requests at a rate of 30–35 packets per second for NDN and CoAP without immediate acknowledgments. Roughly 50% of all requests on both links at the gateway node (see Figure 10.2) consists of request retransmissions. In contrast, separate responses visibly reduce the overall requests from the gateway to below 20 packets per second. Retransmissions represent only ≈ 23% of all requests. The differences are equally pronounced when observing the request to retransmission ratio across the entire network: For CoAP without early acknowledgments and NDN, ≈ 40% of total requests in the network are retransmissions. With the use of early acknowledgments, this number reduces to 25% for CoAP.

All protocols exhibit very similar performance when inspecting the amount of incoming responses at the gateway node. However, CoAP without early acknowledgment shows a subtle decline in overall success rates. Around 96% of requests have a corresponding response, while NDN and CoAP with early acknowledgment both display a success rate of 98%. The reduced number of control overhead does not only reduce the utilization of network resources, but also lessens link stress and increases success rates.

10.5 Discussion

Individual protocol components and their interaction can impact performance significantly. We will now discuss how the exchange of protocol building blocks between the worlds impacts corresponding network performance.

CoAP. We have seen during our journey on deploying a CoAP scenario with ICN characteristics that the combination of two building blocks shows substantial performance improvements. Chaining CoAP proxies with caches enables link-scoped corrective actions. This shortens retransmission paths and reduces link traversals in networks with high loss probabilities. The compact handling of link-local addresses, which can be compressed away, is resource efficient and at the same time demonstrates a formal coincidence with the address-less NDN architecture.

The hop-by-hop forwarding between proxy nodes potentially leaks service paths and therefore sensitive data to the application logic. The problem of name confidentiality is also prevalent in ICN architectures and approaches have already been proposed in the literature that provide obfuscation mechanisms for routed name prefixes [276, 277]. Due to the high similarities between NDN and CoAP proxy deployments, the obfuscating approaches can easily be adapted to the Proxy-URI string components.

Our study also identified that while a retransmission cache is sufficient to gain ICN-like benefits for a single client, content level caching which serves multiple clients not only requires careful application design, but also poses interesting challenges for OSCORE use cases.

Further message size reduction is possible by using the CoAP split options for expressing the URI, and by using reverse proxying styles. Smaller messages are beneficial because of increased transmission success (see Section 10.4.3). Moreover, successful *requests* have the additional effect of building a request path which starts populating caches for later use when responses may be lost (as noted in Section 10.4.5). Our experiments further indicate much higher positive impacts for smaller requests, as they quickly build a request path and profit from hop-wise retransmissions for a response.

As a last point, we want to discuss in-network state for CoAP. The original deployment idea follows the basic packet network concept of stateless forwarding with network state persisting on the endpoints, only. In the information-centric CoAP deployment, all nodes including the forwarders maintain request state. As main memory is constrained in low-power networked devices, the number of open request handles at each node is equally limited. At first sight, the overhead added on each forwarder appears as a disadvantage that may lead to quickly saturating memory resources and denial of service on request paths. Our IoT experiments in NDN deployments, however, show that content caching and request aggregation features are able to limit resource usages immensely by shortening path lengths and reducing completion times of open requests.

ICN. The full CoAP proxy has a similar cache model as CCNx and NDN. Unlike in HTTP, neither protocol family supports cache policy control in request messages. Content producers determine content lifetime values on message creation and requests cannot bypass cache entries en-route. CoAP adds an efficient cache validation model: requests that meet stale cache entries trigger secondary requests to the original server to check on content validity. Returning responses may either include a confirmation of validity or new content. We argue that a cache revalidation model for ICN would optimize bandwidth consumption not only in IoT stub networks and want to pursue its utility in future work.

We see value in adopting separate responses [4, Section 5.2.2] to control the retransmission behavior of previous hops. NDN and CCNx already support an error reporting infrastructure using Interest NACK [278] and Interest Return [19]. These mechanisms could be extended to deploy a similar retransmission control strategy. Adding retransmissions not only to requests, but also to responses is a technique commonly used in CoAP deployments to increase success rates and we suggest an experimental analysis with similar approaches for information-centric deployments.

The CoAP token mechanism seems applicable to reduce response sizes: Requests could carry a short token that maps to a name on each forwarding hop, possibly using the Pending Interest Table (PIT). A similar technique is already employed by the en-route compression functionality of ICNLoWPAN [264]. Instead of mirroring back the full name, responses could include the short token in order to map to the corresponding request on the reverse path. Reducing the response size does not yield the same benefits as reducing the request size, but still reduces a major contributor to the link stress.

10.6 Conclusions

In this chapter, we presented a conceptual feature comparison between CoAP and archetypal ICN designs. We set out with the motivation to build a RESTful CoAP deployment that inherits information-centric properties and conduct a comprehensive analysis to quantify the effective network performances in the (low-power) Internet of Things.

Our findings indicate that (*i*) loosening the end-to-end principle, (*ii*) adding retransmission caches, and (*iii*) utilizing object security enables secure, RESTful deployments that achieve comparable network performances as observed with NDN. As a result of compiling a feature compendium for CoAP and NDN, we were also able to identify striking protocol elements that bear potentials to improve protocol operations if transferred from one architecture to the other.

We have shown that the differences in caching and even in naming between original information-centric designs and those originating from an end-to-end mindset are more by convention than by necessity. Assimilating RESTful CoAP deployments towards a named-data networking architecture allows reusing and exploring the impact of many well-studied concepts in new deployment environments.

Chapter 11

Secure Multiparty Access to Group Content

Abstract

Content replication to many destinations is a common use case in the Internet of Things (IoT). The deployment of IP multicast has proven inefficient though, due to its lack of layer-2 support by common IoT radio technologies and its synchronous end-to-end transmission, which is highly susceptible to interference. Information-centric networking (ICN) introduced hop-wise multiparty dissemination of cacheable content, which has proven valuable in particular for low-power lossy networking regimes. NDN, however, the most prominent ICN protocol, suffers from a lack of deployment.

In this chapter, we explore how multiparty content distribution in an information-centric Web of Things (WoT) can be built on CoAP. We augment the CoAP proxy by request aggregation and response replication functions, which together with proxy caches enable asynchronous group communication. In a further step, we integrate content object security with OSCORE into the CoAP multicast proxy system, which enables ubiquitous caching of certified authentic content. In our evaluation, we compare NDN with different deployment models of CoAP, including our data-centric approach in realistic testbed experiments. Our findings indicate that multiparty content distribution based on CoAP proxies performs equally well as NDN, while remaining compatible with the established IoT protocol world of CoAP on the Internet.

11.1 The Problem of Multicast in the IoT and Related Work

11.1.1 Challenges of IoT Group Scenarios

IoT message exchange follows the patterns ‘scheduled’ and ‘on demand’ between individual node pairs or in groups. Group communication is desired for data fusion, *e.g.,* when a group of sensors returns its readings following a single subscription or data request. Group communication is also needed for disseminating data to large sets of receivers, *e.g.,* for distributing instructions to actuators, and may consist of large data volumes, for instance in the case of software updates (see Figure 11.1).

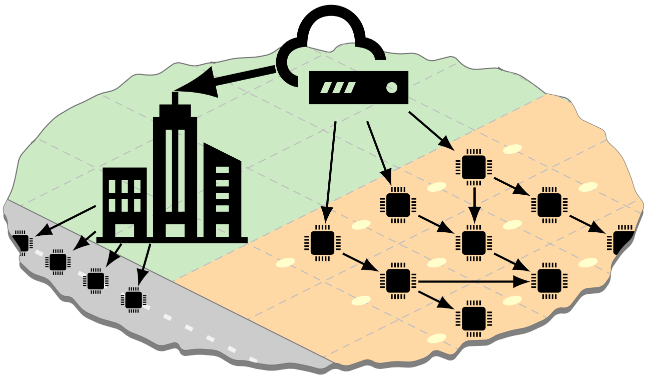


Figure 11.1: Massive firmware roll-outs in distributed and heterogeneous networks.

Group communication requires scalable network solutions, whenever an iterated unicast transmission between participants will impose critical stress onto network links. Such limits are quickly reached in low-power lossy wireless regimes for which rates of 100 packets/s may already strain a link. With battery-operated or energy-harvesting devices, energy conditions are often even more critical. Traffic flows greatly dominate energy expenditures [279] and hence the lifespan of the involved nodes.

11.1.2 Multicast and Its Limitations in the IoT

The traditional approach to group communication on the Internet is IP multicast [280]. Multicast network costs scale well as a root of the number of receivers [281], and IP multicast seamlessly hosts stateless UDP transport, which dominates the IoT. Further, mappings exist to common link layers of local area networks such as Ethernet or WLAN.

Constrained wireless technologies, however, that do not employ any form of slotted channel access, but rather use carrier-sensing (*e.g.,* IEEE 802.15.4 [2]) or pure ALOHA (*e.g.,* LoRAWAN [282]), typically lack support for multicast on the link-layer and default to broadcast. Also for Bluetooth Low Energy (BLE) [283], which utilizes proper multiplexing schemes by the frequency and time domain to reduce radio listening cycles, an efficient IP multicast mapping is not given. Since BLE connections between devices are point-to-point, IP multicast is realized by duplicating multicast messages on each unicast link [284, Section 3.2.5].

Multicast routing in the Internet is complex and multicast mobility adds further complications [237]. Bit Index Explicit Replication (BIER) [285] is a new multicast architecture that promises a considerable simplification by eliminating per-flow state from routers. The Multicast Protocol for Low-Power and Lossy Networks (MPL) [286] establishes IP multicast forwarding in constrained and wireless deployments. Instead of building a dissemination tree, MPL uses a controlled flooding approach combined with a mechanism to detect and suppress the propagation of duplicate messages. Other approaches [287, 288] supplement the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [202] to operate in a similar fashion as BIER.

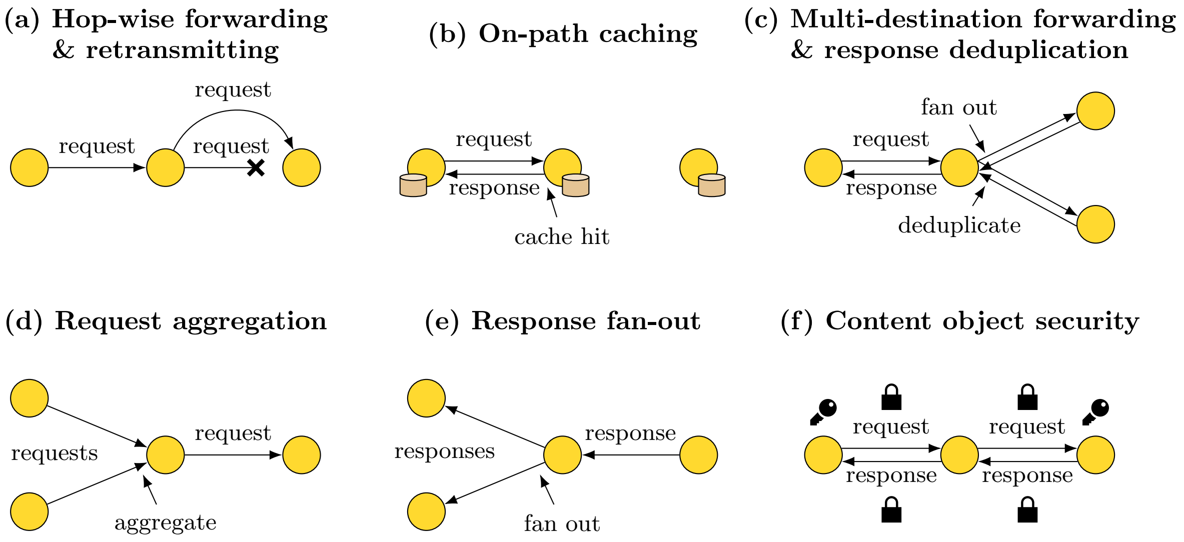


Figure 11.2: NDN protocol features that enable an efficient and secure multiparty communication for the IoT.

CoAP extensions [243, 289, 290] update the request-response model to enable one-to-many communication using multicast IP addresses for resource endpoints in the application layer. While this addition allows clients to perform requests to a group of CoAP servers, the resulting responses always return as unicast to the respective client. To remain stateless, a CoAP group communication only allows non-confirmable multicast requests [289, Section 2.3.1].

The protection of communication flows between multiple endpoints poses another challenge. Transport layer security is the default strategy to deliver protective measures for streams in the Internet [60] and for datagrams in the IoT [25] between two hosts. Security contexts are tightly bound to socket endpoints and use the five-tuple (IP$_{src}$, Port$_{src}$, IP$_{dst}$, Port$_{dst}$, Transport) to identify secured channels. This strong binding makes transport layer security impractical in flows involving multiple devices.

11.2 Multiparty Content Retrieval for a Data-centric Web of Things

11.2.1 Named-Data Networking for the IoT

Named-Data Networking (NDN) [21] is a future Internet architecture that follows the information-centric networking (ICN) paradigm. NDN employs a stateful and pull-driven forwarding fabric with built-in reliability features and multicast support, which focuses around the request-response communication pattern. In comparison to the general-purpose Internet, where only a single IP datagram exists to encapsulate upper layer packets, NDN specifies two message types on the network layer with contrasting semantics: *Interests* for requesting content and *Data* for delivering responses. NDN exhibits protocol features (see Figure 11.2) that have been proven valuable in constrained and wireless IoT networks [291, 98, 111].

11.2.2 NDN Protocol Features

Hop-wise Forwarding & Retransmitting. Interest messages traverse hop-by-hop and are forwarded on human-readable names akin to URLs for web resources. Each forwarder tracks state for open requests in the *Pending Interest Table.* Data messages return on the constructed request path and consume the existing forwarding state. When a response is lost as illustrated in Figure 11.2 (a), hop-wise timeouts occur and requests are retransmitted. Retransmissions are confined to links that have not been traversed by responses yet.

On-path Caching. Individual hops maintain a content store as an integral part of the forwarding logic. Data are stored in this cache and returned for matching requests as displayed in Figure 11.2 (b). Caching provides location-independence for content and is a fundamental feature to improve bandwidth and latency of content retrievals. Especially in low-power setups with intermittent connectivity, caching paired with the previously discussed corrective behavior yields an increased robustness due to shortened request paths on retransmissions.

Multi-destination Forwarding & Response Deduplication. The Forwarding Information Base (FIB) records content names and outgoing faces. One compelling difference to an IP FIB is that multiple faces can exist for a single destination to enable one-to-many flows (see Figure 11.2 (c)). A forwarding strategy commits to either all outgoing faces at once, or makes sensible decisions to select a subset. If multiple responses return as a result of request fan-outs, then they are deduplicated.

Request Aggregation & Response Fan-out. Simultaneous requests from multiple origins can meet on shared paths as shown in Figure 11.2 (d). Messages aggregate on hops that previously set up appropriate forwarding state. Incoming faces of equal requests are cataloged and when a response returns, it fans out to all stored faces (see Figure 11.2 (e)).

Content Object Security. Unlike schemes that focus on transport layer security between endpoints, messages are cryptographically signed and content can be encrypted without requiring any endpoint information (see Figure 11.2 (f)). Content object security allows content caching for long periods of network disruption, while preserving all security measures without maintaining endpoint-based security contexts [292].

11.2.3 An Information-centric Web of Things

In previous research, we designed an information-centric Web of Things (WoT) [105] that uses CoAP forward proxies on every hop along a path. This deployment embodies a subset of the NDN protocol features displayed in Section 11.2.2. As their comparative measurements show, a hop-wise forwarding, hop-by-hop retransmissions, and response caching on each proxy node are sufficient to boost the network resilience up to the level of standard NDN setups.

However, their architectural design misses these integral features to enable an efficient multiparty communication: (*i*) multi-destination forwarding with response deduplication, (*ii*) request aggregation from multiple origins with response fan-out, and (*iii*) a pluralistic cache utilization. Their naïve OSCORE [26] integration provides content object security, but confines the effects of caching and request aggregation only to request-response pairs due to the strong message binding that OSCORE introduces. With this limitation, only retransmissions benefit from information-centric properties.

In the remainder, we will extend the information-centric WoT construction with support for multiparty content access by integrating the missing features into the CoAP deployment.

11.2.4 Multi-destination Forwarding

When a CoAP node attempts to obtain a resource representation, it encodes the resource URI in a set of CoAP options, either in the *Proxy-Uri* option or using individual options (see Figure 11.3). Note that while the orange part is named *host* due to its predominant use, it may use any locally defined lookup system [293, Section 3.2.2].

$$\underbrace{\texttt{coap}}_{\text{Proxy-Scheme}}\texttt{://}\underbrace{\texttt{n1.example.com}}_{\text{Uri-Host}}\texttt{/}\underbrace{\texttt{temp}}_{\text{Uri-Path}}\texttt{?}\underbrace{\texttt{t=12345}}_{\text{Uri-Query}}$$

Figure 11.3: Equivalent components of a Proxy-Uri as used with forward proxies.

Nodes may send a request to the known network address of the server, or to a usually more powerful proxy node to delegate routing, name resolution, and protocol conversion complexity. In the common case of deferred routing, the proxy will use DNS and its routing table to send the request to the server.

In the information-centric WoT design (see Section 11.2.3) forward proxies are used on each hop along a path, and the Proxy-Uri option is present as a forwarding hint in all hop-wise requests but the last. On that final hop, the scheme and host information can be discarded, and the more compact Uri-Path and Uri-Query are sent instead of Proxy-Uri.

A FIB structure (see Table 11.1) managed on the proxy stores the next-hop, and whether that hop requires a full Proxy-Uri option including the *host* component. Analogous to NDN, we adjust this FIB to allow multiple next-hop entries for a single destination. If multiple next-hops exist, the request is duplicated to all available endpoints. As CoAP requests are not necessarily idempotent (some are not even side-effect free), requests with codes like POST or PATCH still have to take a single next-hop.

URI pattern	Next-hop	Send host
coap://00-01/temperature*	coap://[fe80::1%0]	yes
	coap://[fe80::2%1]	yes
coap://00-02/firmware/*	coap://[fe80::3%0]	no

Table 11.1: Application-level Forwarding Information Base for CoAP.

11.2.5 Response Deduplication

Duplicate responses return to a forwarder node if requests are replicated onto different paths as part of the multi-destination forwarding process (see Section 11.2.4). A forwarder has two choices when tasked with the deduplication of response messages. First, if content is dynamic and not bound to the resource path, then all returning responses are aggregated to form a single response. An in-network deduplication, however, has the issues of deducing correct delays to capture all returning messages and to apply a reduce function of arbitrary complexity depending on the use case. The second choice consists of forwarding only the first arriving response message, while discarding all other occurrences. This case is more efficient, because it reduces link stress to a minimum, but is only effective in deployments where content binds immutably to resource paths. In our design considerations for an information-centric architecture, we opt for the second choice as it is conforming with the philosophy of NDN to have a strong name to content binding.

In CoAP, responses match to corresponding requests with the use of token values. A client generates a token and includes it in a request. To make the required CoAP state for requests similarly compact as in NDN, clients can use a shared token space for all their next-hops. Then, incoming responses consume that request state, and get forwarded to all interested clients. Any further responses based on that token are rejected or ignored as illustrated in Figure 11.2 (c).

11.2.6 Request Aggregation and Response Fan-out

Simultaneous requests to the same resource path from different clients can be aggregated to reduce link stress. In parallel to checking for cached responses using the CoAP Cache-Key [4, Section 5.6] based on CoAP options, the set of open request states is checked using the same key. If state exists, then a forwarder records the newly arriving request along with the existing information. Tokens in client requests provide a unique identification for transactions and must be stored individually to preserve the request-response matching as illustrated in Figure 11.4.

For the very first request, the forwarder sends out a new request message that is semantically equivalent to the original request. Tokens (along with some other message properties) are local to the hop, and thus generally differ between incoming and forwarded requests. When a

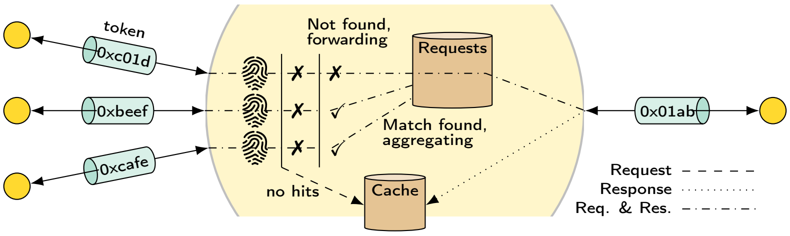


Figure 11.4: Forwarding logic for request aggregation and response fan-out.

response returns, the list of interested clients contains their stored addresses and tokens, from which corresponding responses are built and fanned out.

11.2.7 The Problem with Content Object Security

The information-centric WoT deployment leverages OSCORE [26] to protect CoAP messages across network boundaries. It provides request and response confidentiality, integrity across request and response, and source authentication even in its group mode [244]. A COSE [240] object is populated with a statically preconfigured or dynamically derived [294] OSCORE context. While a COSE header holds meta information, *e.g.*, key identifiers and encryption algorithm, a COSE ciphertext contains parts of a CoAP message that are considered for encryption. A protected CoAP message then carries the COSE header as an OSCORE option and the COSE ciphertext as payload (see Figure 11.5).

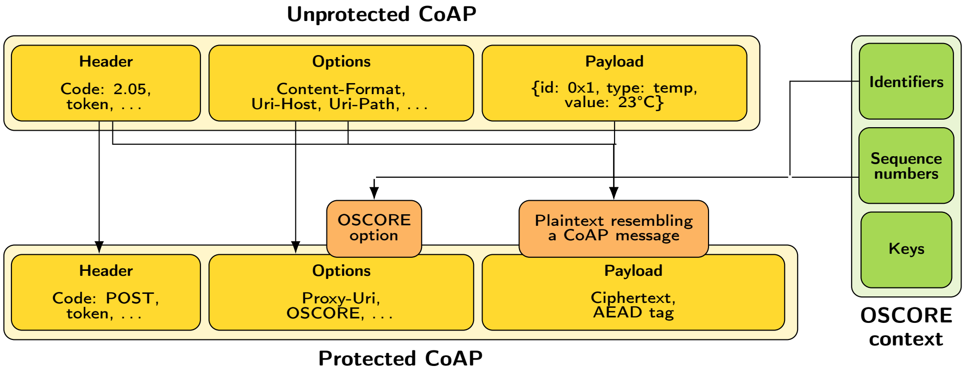


Figure 11.5: COSE and OSCORE to protect CoAP messages on the object level.

OSCORE complements the information-centric WoT with security on the content object level to enable secured CoAP deployments that are on par with NDN setups in terms of network performance. There are, however, two aspects that greatly affect the carefully designed multiparty content access functionality when OSCORE is naïvely employed on request paths that solely consist of proxy nodes: (*i*) reduced cache utilization and (*ii*) impaired routing decisions.

Caching. Clients generate unique nonces that are used by cryptographic operations for request-response pairs. With the nonce both directly (in the OSCORE option) and indirectly (in the ciphertext) altering the Cache-Key, distinct OSCORE requests can not share caches. Even if cache hits could produce an equivalent earlier response, the request-response binding would fail to perform the authenticated encryption with associated data (AEAD) decryption. The only exception where caching OSCORE messages shows effective results is for request retransmissions on packet loss [105].

Our architectural decision for re-activating the caching support is to use deterministic requests [261]. It uses OSCORE group communication and introduces the deterministic client, a fictitious group member which avoids nonce reuse not by using sequence numbers in nonces, but by hashing the request plaintext and additional data into the key. Any member of the group can assume the role of the deterministic client by asking the group manager for the correct key details. Requests with identical plaintext sent from any group member results in the same hash and identical ciphertext. These requests carry the full hash in a dedicated CoAP option, which is also included in the calculation of the Cache-Key. A server receiving such a request uses the sent hash to derive the cryptographic keys, decrypts, verifies the hash from the plain text, and responds in group mode for any group member to use the response.

Deterministic OSCORE weakens three properties of OSCORE: (*i*) the request-response binding (responses are only bound to *a* request of the same content, not *the* request the client created), (*ii*) source authentication for requests (which is tolerable as the mechanism only applies to side-effect free requests), and (*iii*) request confidentiality (but only to the extent that an adversary can see that two requests are equal). Consequently, deterministic requests bypass the replay protections of OSCORE, which is not a significant issue if only idempotent requests are used. Deterministic OSCORE preserves source authentication for responses by using asymmetric cryptographic signatures.

Routing Decisions with Request Confidentiality. OSCORE prescribes [26, Section 4.1.3.] a different treatment for options that form the request URI (see Figure 11.3): while path and query parameters are encrypted and thus privacy protected, scheme and host are neither encrypted nor integrity protected. For the latter, an implicit protection is in place: The cryptographic context choice of the client ensures that only the right server can respond. This reflects the HTTP practice that any request to the same origin (*i.e.,* scheme, host and port) is protected in a single connection. It also enables a flexible application layer routing using forward or reverse proxies. For name-based FIBs like in NDN, this severely limits the information available for forwarding decisions unless the forwarder is a group member. Information can still be used in hash based path choices (*e.g.,* to load balance across several larger caches) but not by expectations of response characteristics like in Table 11.1.

As the proposed WoT construction does not depend on the host component to map directly to a host, there is room for application designers to move information between the host and path components. They can put more information into the unencrypted host component (in the Table 11.1 example, use coap://00-02-firmware/ instead of coap://00-02/firmware/) and thus expose more request information, or move more information into the path component

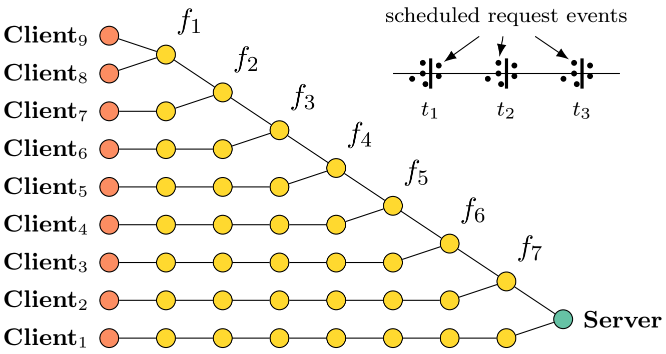


Figure 11.6: Testbed topology modeling fan-outs from rank zero to seven of the forwarding hierarchy.

(*e.g.*, coap://example.com/00-02/firmware/) and thus hide more information at the cost of less configurable routing. In this extreme form, nodes are practically required to be group members to make meaningful routing decisions.

11.3 Experimental Evaluation

We experimentally assess the protocol performance of the information-centric WoT in a multi-party content retrieval scenario and compare it against NDN using real IoT hardware.

11.3.1 Experiment Setup

Scenario

We experimentally assess the protocol performance of the data-centric WoT in a typical command-and-control scenario where multiple actuators show interest in up-to-date instructions from a control node. For our evaluations, we construct the network topology as depicted in Figure 11.6. The gradual addition of intersections and the long path stretch allow for observing protocol behaviors and inspecting the performances in a better nuanced event space. A set of nine clients connects to a server that periodically assesses sensory data and the environmental situation to generate timely instructions every second. In turn, all clients request the latest instruction by appending an increasing sequence number as time offset to the resource name: `/instruction?t=`x. At experiment begin, we synchronize each client to start the scheduled request pattern simultaneously and clients apply a random jitter in the hundreds of milliseconds range to each request. In total, each client triggers 1000 requests.

The path between client$_9$ and server demonstrates the most intersections and it is reasonable to assume the highest traffic load for this route. All forwarders on this path are named $f_{1\text{–}7}$ for easier reference in the evaluation.

Hardware and Software Platform

We conduct our experiments on the `grenoble` site of the IoT-LAB testbed [27]. It provides a large multi-hop deployment consisting of various class 2 [1] devices featuring a 32-bit ARM Cortex-M3 μcontroller with 64 kB of RAM and 512 kB of ROM. The platform further mounts an Atmel AT86RF231 [69] transceiver to operate on the 2.4 GHz IEEE 802.15.4 radio. Devices run the RIOT [28] operating system in version 2021.01. OSCORE experiments use `libOSCORE`[1] and the modular GNRC IPv6 network stack, while the NDN setup uses CCN-lite [106].

Protocol Settings

We compare four different protocol deployments with varying degrees of multiparty support and summarize their features in Table 11.2.

i) OSCORE: end-to-end CoAP deployment that uses OSCORE to protect single request-response transactions.

ii) OSCORE Proxy: hop-by-hop CoAP deployment protected by OSCORE. Response caching and request aggregations are confined to request-response pairs only.

iii) Deterministic OSCORE Proxy: hop-by-hop CoAP deployment with deterministic requests and OSCORE protection. Caching and aggregation work across multiple request-response pairs from varying endpoints.

iv) NDN: hop-by-hop NDN deployment with caching and aggregation working for multiple requests and endpoints.

Protocol	Caching	Request Aggregation	Response Fan-out
OSCORE	—	—	—
OSCORE Proxy	single party	only retransmissions	—
Det. OSCORE Proxy	multiple parties	multiple parties	✓
NDN	multiple parties	multiple parties	✓

Table 11.2: Multiparty characteristics of the selected deployments.

We configure three frame retransmissions with an exponential backoff in the range of milliseconds in case of missing acknowledgments on the link-layer. Respectively, we enable three request retransmissions for CoAP and NDN using a message timeout of two seconds. Request buffers are sufficiently dimensioned to not reject retransmitting requests due to unavailable buffer space.

[1] https://gitlab.com/oscore/liboscore

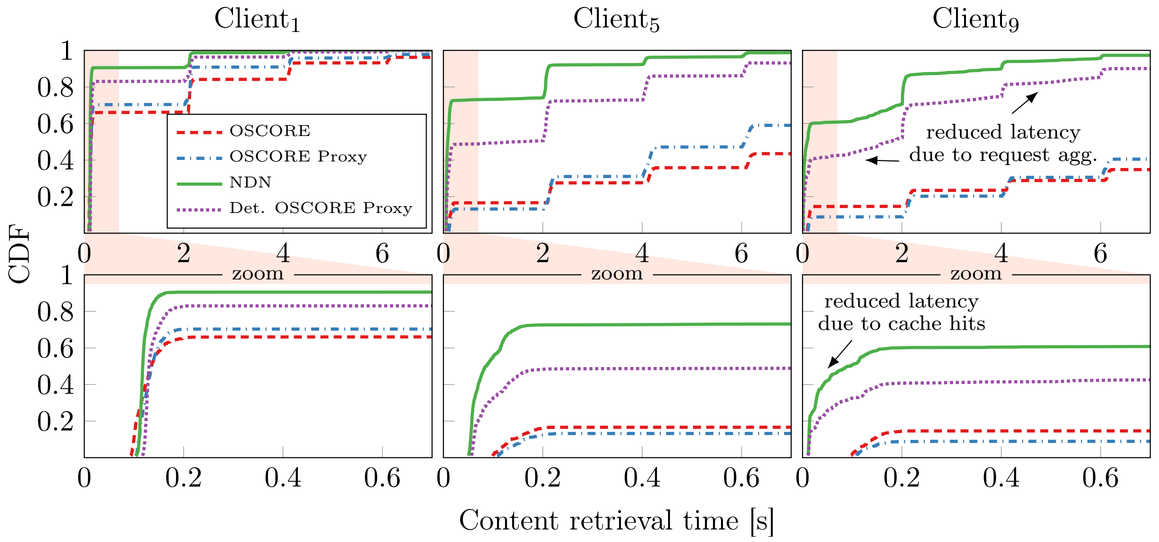


Figure 11.7: Content retrieval times—distributions for different client devices and protocol deployments.

Cache implementations for CoAP and NDN are equivalently scaled to hold 40 responses, which is an adequate number for our setup to not replace up-to-date cache entries required by delayed retransmissions.

Security Configuration

The standard OSCORE operation provides peer authentication with a strong message binding between single requests and responses. To preserve the peer authenticating characteristic of OSCORE in a group-key environment, deterministic OSCORE leverages digital signatures in responses using the Edwards-curve Digital Signature Algorithm (EdDSA) that is carried inside the response payload. This ensures that members can access and read encrypted messages, but the digital signature prevents forgery attempts by group members. Software based signature derivations can occupy the processing unit by more than a second, which renders a realistic content retrieval scenario with multiple parties and prominent traffic patterns unusable. Since the hardware platform in the testbed does not provide hardware-assisted crypto operations, we replicate the signature derivation time of common cryptoprocessors and set it to a constant-time delay of 20 ms [232] in our server firmware.

11.3.2 Comparative Evaluation

Content Retrieval Time

In our first evaluation, we gauge the time from initiating a request on a client to the arrival of the requested content on the application. This time measurement does not only include pure message round-trips, but additionally scales with packet loss and retransmission events. Figure 11.7 summarizes the distributions of content retrieval times for our protocol ensemble and clients$_{1,5,9}$ from Figure 11.6.

We observe that all deployments are in agreement with the retransmission behavior. Successful message deliveries without corrective actions finish in the sub-second range, while retrieval times multiply by the configured retransmission timeout into the seconds range on packet loss. This leads to a staircase pattern every two seconds for all distributions. CDF maxima for each protocol marks the overall success rate.

Client$_1$ shows an overall positive success rate for all protocol expressions: more than 96% of requests succeed for the end-to-end OSCORE deployment, while the hop-by-hop variants increase up to 100%. Since client$_1$ retrievals are not affected by cross traffic from other clients, these affirmative results are expected. The distributions progressively degrade for each client if we move towards client$_9$ due to the increased traffic load that manifests on the shared path between f_1 and f_7. Three aspects become visible when we observe the completion time distributions for client$_9$. First, the multiparty-unaware deployments display reduced success rates of up to ≈40%. OSCORE Proxy minimally improves on the standard OSCORE setup because of cache hits, which are however confined to request retransmissions from a single client. NDN and the deterministic OSCORE Proxy deployment open up this restriction and allow the utilization of cached responses by all clients. This leads to steady success rates across all client nodes independently of the induced traffic load. Second, the latency in the sub-second range improves drastically from around 100 ms down to ≈25 ms for the multiparty-aware deployment variants due to the availability of prepopulated caches. Third, the latency for request retransmissions in the seconds range equally reduces for NDN and the deterministic OSCORE Proxy variant as displayed by the steep staircases. This results from request aggregations on hops that experience loss events and have in turn already scheduled retransmission events. Returning responses then fan out to all interested clients prematurely well before local retransmission timeouts.

Cache Utilization and Request Aggregation

In our next comparison, we measure the server load for our protocol selection to quantify the effects of cache utilization and request suppression. Figure 11.8 presents the response transmission rate on the server node for the duration of the experiment.

In accordance with the number of clients and the configured scheduling interval for requests, the optimum rate on the server is 9 packets/s in deployments without cross-client caching. We observe an approximation of this expected value for the plain OSCORE and OSCORE Proxy setups. In the first case, the average transmission rate is above ten, which is evidence of request retransmissions, and in the latter case, the rate averages to slightly below the optimum value. This comes as no surprise, since retransmissions of the same origin can benefit from cache hits in the network. Requests that never arrive at the server result in a slightly reduced average.

The optimum average rate scales down to 2 packets/s when caching and request aggregation

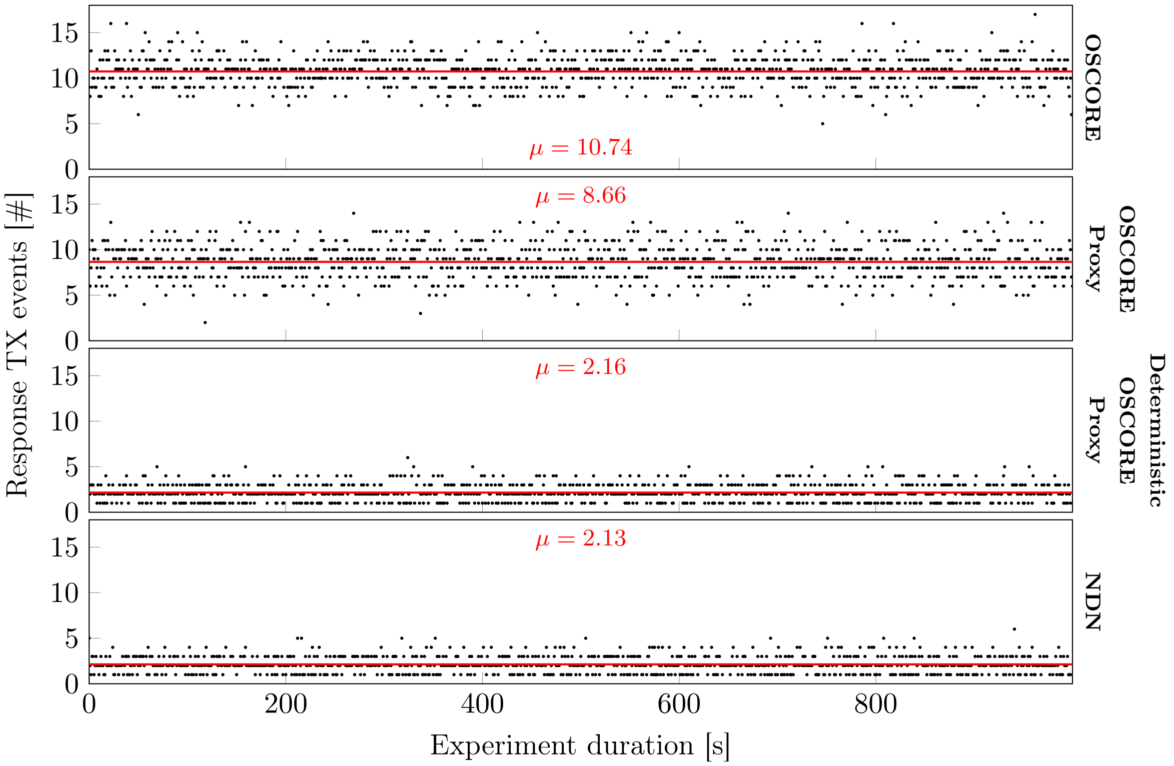


Figure 11.8: Quantity of outgoing responses measured at the server node.

is enabled. When precisely scheduled, all requests in the subtree containing clients$_{2\text{–}9}$ collapse into a single request that reaches the server and the additional request arrives from client$_1$. Deterministic OSCORE Proxy as well as NDN show in Figure 11.8 an average transmission rate that is close to the theoretical optimum.

Security Effort

To put computational effort associated with the different protocols into perspective, we compare the number of performed and necessary cryptographic operations. Numbers in Table 11.3 express operations per successful completion of a request by a single client. The ideal conditions against which we benchmark are derived from the analysis of the configured topology in Figure 11.6 assuming no packet loss.

For the OSCORE deployment, the optimal number of AEAD computations is two for clients and the server. Clients perform an AEAD operation when they generate a request and when they receive a response. Conversely, servers operate similarly for receiving a request and generating a response. We observe in Table 11.3 that on average clients perform 47% more AEAD operations when compared against successfully received responses. This is a result of packet loss, where retransmissions were not able to recover lost response messages. Note that request retransmissions do not add to the overhead, because they already reside in the retransmission

Operations	OSCORE		OSCORE Proxy		Det. OSCORE Proxy		NDN	
	Client	Server	Client	Server	Client	Server	Client	Server
Authenticated Encryption	2.9(↑47%)	4.6(↑132%)	2.7(↑33%)	3.2(↑61%)	2.0(↑4%)	0.5(↑11%)	1	0.2(↑5%)
Signature Creation / Verification	—	—	—	—	1	0.2(↑9%)	—	—
Message Authentication Code	—	—	—	—	3.2(↑8%)	0.7(↑9%)	1	0.2(↑5%)

Table 11.3: Mean cryptographic operations per successful content retrieval. In parenthesis, the relative overhead added by packet loss.

buffer with security related information populated. The server side shows a computational effort that increases by 132%. Arriving request retransmissions lead to multiple generations of a response.

The OSCORE Proxy deployment behaves equally with respect to the necessary cryptographic operations. Due to the caching and aggregation features that only span a single request-response pair, message losses are reduced. Effectively, this decreases the amount of unnecessary AEAD operations for clients and the server when compared to OSCORE.

In the deterministic OSCORE Proxy, the averaged AEAD operations further decrease due to the multiparty-aware protocol operation. The high success rates on clients yield a nominal AEAD overhead of only 4%. Given our topology in Figure 11.6 and the configured request scheduling, only two of nine requests reach the server per second in ideal conditions. Requests arriving from the upper subtree collapse into a single request during forwarding and another single request arrives from client$_1$. The amortized ideal number of AEAD operations on the server is thus $2 \cdot \frac{2}{9} = 0.44$. In addition to AEAD operations, this deployment requires the creation and verification of digital signatures in responses. Clients have an optimal value of one signature verification for each successfully received response. The server creates $\frac{2}{9} = 0.22$ signatures per request round for all clients. The very low number of retransmissions that actually arrive at the server produce an extra effort of 9%. This deployment further uses three HMAC operations per request and response. Clients demonstrate analogously a small overhead and the server equally shows an added effort of 8% from the optimal value $3 \cdot \frac{2}{9} = 0.67$.

The NDN deployment presents comparable overheads. The ideal number of operations differ significantly, though: In this setup, only responses are signed and carry AEAD encrypted content.

For a useful comparison of OSCORE with deterministic OSCORE, the added asymmetric

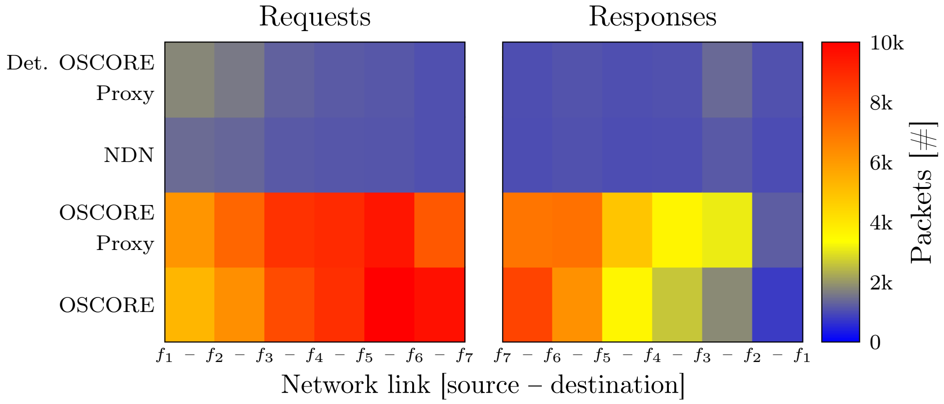


Figure 11.9: Requests and responses recorded along the shared path $f_{1\text{–}7}$.

operations need to be considered with their time or power consumption; the former outweighing the latter by a factor of over 100 in both aspects [232]. Taken on their own, the energy savings of caching fail to justify the use of deterministic OSCORE with this chosen topology. However, considering the combined energy use of cryptographic operations and radio transmissions, where the consumption of a frame transmit plus receive operation (about $10^{-3}J$) is larger than that of even an asymmetric cryptographic operation (about $0.5 \times 10^{-3}J$), it is easily seen that the reduced link utilization of deterministic OSCORE amortizes the increased energy cost of the asymmetric cryptography.

Link Stress

Our protocol collection shows nuanced variations in the amounts of packet transmissions throughout the topology. Figure 11.9 illustrates absolute packet transmissions for the path between forwarder nodes $f_{1\text{–}7}$. In general and given the intersections of our constructed tree topology in Figure 11.6, the amount of request transmissions increases gradually for each hop that is closer to the server. On the other hand, the amount of transmitted responses between forwarders naturally decreases when moving away from the source.

Each client triggers 1000 distinct instruction requests, *i.e.*, every forwarder f adds another 1000 packets to the upstream path when there is no packet loss in addition to the packets received downstream. For the OSCORE and OSCORE Proxy configurations, we observe the anticipated continuous increase in requests. OSCORE Proxy shows slightly more requests per link, because retransmissions take place hop-by-hop, while there is a much higher chance of losing request retransmissions along the path for the end-to-end OSCORE. This characteristic yields insignificantly less link stress for OSCORE, but does not promote success rates as we have observed in earlier evaluations. Downstream transmissions show similarly expected results: Unnecessarily large quantities of returning responses appear close to the server and this number declines very fast with each hop for OSCORE due to packet loss. OSCORE Proxy smooths out the steady decrease in responses with the use of response caching for retransmissions of the original request. The multiparty-aware protocols operate comparably with small differences resulting from larger packet sizes for the deterministic OSCORE deployment. The pluralistic cache utilization and request aggregation features allow both variations to reduce the packet transmissions on each forwarder hop to approximately 1000–2000 packets.

11.4 Conclusions

Information-centric content replication has repeatedly proven beneficial in low-power networks. Hop-by-hop forwarding, in-network caching, and hop-wise retransmissions are key promoters of reliably delivering packets in lossy wireless regimes. This work is part of an ongoing effort to develop an information-centric Web of Things (WoT) that is built on CoAP and OSCORE. In this chapter, we designed protocol configurations and extensions that carry one-to-many CoAP data flows with OSCORE content object security.

In detail, we first explored missing multiparty protocol features, then identified protocol entry-points for our extensions, and finally laid out a blueprint for integrating (*i*) multi-destination forwarding with response de-duplication, (*ii*) request aggregations with response fan-outs, and (*iii*) a pluralistic cache utilization.

Using full protocol implementations on RIOT OS and a real-world testbed, we comparatively evaluated our multiparty communication model against CoAP OSCORE (w/ and w/o Proxy) and NDN. We found significant improvements in network utilization and a much reduced link stress. On the overall, we could show that multiparty content dissemination works equally efficient with our CoAP-based WoT solution and NDN. The results confirm that information-centric principles can be built into the CoAP ecosystem without sacrificing interoperability nor performance.

Chapter 12

Conclusions and Outlook

Information-centric principles promote an efficient replication of content. Initially designed for resource-abundant and general-purpose, wired networks, Information-Centric Networking (ICN) also offers many benefits for the low-power, resource-constrained Internet of Things (IoT), including a reduced complexity of network stacks, loose coupling between content and producers, and a hop-wise retrieval of data using content caches on the network layer. Nevertheless, protocol integrations might conflict with the limited processing power and reduced memory capacities of various IoT hardware platforms. These limitations challenge network scalability and the operational lifetime of battery-operated devices. This thesis comprehensively addressed research questions that relate to ICN in the IoT and the application of ICN concepts in the emerging Web of Things (WoT). Part I focused on deepening the understanding of information-centric principles in lossy, wireless networks to design an efficient ICN protocol integration for low-power regimes. Part II discussed a realistic deployment trail using standard IoT technologies. The following list summarizes the contributed solutions, evaluative testbed experiments, and deployment experiences for the individual research areas from Section 1.3 in more detail.

Protocol Behaviors in Harsh Environments. As a first step to this research agenda, important industrial use cases and requirements backed up by field experiences of the safety-critical industry were characterized and summarized in Chapter 3. Thorough comparative analyses based on extensive real-world experiments and the previously identified requirements were performed on actual IoT devices. These experiments compared the three common protocol families CoAP, MQTT-SN, and NDN in dense scenarios of 50 constrained nodes arranged in a multi-hop topology, and the implementations were made publicly available to provide an evaluation framework for protocol assessment in the Industrial IoT (IIoT). The results demonstrated that lean push-based protocols operate well in relaxed environments, while pull-based schemes struggle with latency constraints of unscheduled alerts. Under stressed network conditions, however, NDN and HoPP showed a superior network performance, reduced latency, and improved bandwidth utilization compared to the opposing IP-based CoAP and MQTT-SN protocols.

Link-layer Convergence. In the information-centric IoT deployment, ICN protocols run on top of the link layer to reduce stack complexity. This demands a proper link convergence since ICN transmissions are not optimized for low-power and lossy links. Chapter 4 introduced ICN-

LoWPAN, a full-featured protocol convergence layer for an information-centric IoT. ICNLoWPAN adapts NDN and CCNx to the Low-Power Wireless Personal Area Network (LoWPAN) edge by integrating them with compression, framing, and fragmentation. The framing and fragmentation components were aligned with 6LoWPAN to enable the coexistence of ICN protocols and IPv6 deployments in the same LoWPAN. The design considerations for the compression scheme could leverage the benefits of the stateful forwarding. While a stateless compression decreases the verbosity of distinct message headers, a stateful compression reduces redundancies between request-response pairs. Evaluative experiments on real IoT hardware revealed a significant packet size reduction, a decreased energy consumption, and an increased reliability.

Quality of Service. System resources are generally limited in (often) under-provisioned IoT networks. Buffer spaces in the network stack can quickly saturate, which then degrade the overall network performance. A proper balancing of these system resources is, thus, necessary to maintain the efficacy of information-centric content retrievals. Chapter 5 thoroughly explored the impacts of QoS on NDN in the resource-constrained IoT, and designed a simple resource management scheme. While QoS in the IP world is mainly restricted to managing link capacities and packet queues, NDN offers additional resource dimensions such as in-network caches and pending request states that can shape the network performance significantly. This QoS scheme employed the NDN resources, correlated them internally on a node, and further externally between nodes without additional signaling overhead. Fairness measures that prevent network members from starvation further have been designed and analyzed. Extensive experiments in a large testbed confirmed the efficacy of this approach. These QoS measures did not sacrifice the performance of unprioritized traffic, best-effort flows did still uphold their performance or even improve, and correlating the resources showed positive effects, which raised the overall network performance to a higher level than in the state of uncoordinated resource allocations. These results clearly strengthened ICN as a candidate for differentiated IoT network services.

Producer-mobility and Delay-tolerance. Many IoT use cases, especially in industrial settings, require network robustness as well as increased resiliency to producer mobility and intermittent link connectivity. While the information-centric content retrieval already displays a loose coupling, further measures are desired to address mobility and network disruption. Chapter 6 presented a lightweight publish-subscribe system to augment NDN deployments with an additional decoupling in time and synchronization. Experimental evaluations in large, realistic testbeds with varying topologies confirmed that HoPP (*i*) overcomes the complexity of routing named data objects by publishing content hop-wise along a gradient towards content proxies, and (*ii*) ensures an increased reactivity with a low route repair overhead; (*iii*) HoPP proved to retain a continued operation after long handover delays, or even complete network partitionings.

Motivating Reliable Content Distributions in the WoT. One use case that makes the secure and reliable distribution of large content objects necessary for the emerging Web of Things (WoT) is the roll-out of firmware updates. Chapter 8 devised and evaluated procedures for scalable software updates in the constrained IoT with a focus on security and reliability. The target objective was to showcase the effects and benefits of information-centric principles for massive firmware roll-outs, and to motivate a continuing effort towards integrating these principles into the host-centric IoT. The key contributions were (*i*) context-specific naming, version discovery, and verification, (*ii*) scalable and reliable chunk distribution across nodes with inbuilt DoS detection, and (*iii*) thorough experimental evaluations of different update strategies in a real testbed with realistic multi-hop radio links. Results demonstrated that rapid roll-outs in deep multi-hop topologies will fully exhaust resources, whereas the contributions lead to a slower, cascading update strategy, which leaves sufficient resources at intermediate nodes for continued operations.

Security Model for the IoT. Content object security is an orthogonal approach to secure communication on the Internet, which changes the session-centric paradigm by adding authentication and encryption (if desired) to each data chunk—independent of its endpoints. In turn, it allows for content caching and transport translation at gateways, while preserving all properties of data security. Chapter 9 revisited the current competing IoT secure protocol architectures, and presented the full solution space for securely exchanging content objects. The contributions include qualitative evaluations on the prospects and potentials of content object security as deployed by NDN and CoAP with OSCORE, and comparisons with CoAP deployments protected by transport layer security. Findings indicated that in challenging environments protocols with a hop-wise transfer and caching of secured content objects decidedly outperform protocols that build on a protected, end-to-end transport service.

Information-centric Principles for the WoT. Previous assessments prove the advantages of data-centric approaches for low-power networks. Chapter 10 examined the emerging building blocks of the CoAP protocol suite to answer the question whether a restful WoT can be built that adheres to ICN first-hand principles and performance. Each CoAP protocol element was carefully explored in quality and quantity in comparison to NDN, and discussed how they can contribute to an information-centric IoT. The performance of various scenarios that range from a plain CoAP deployment over several extended settings including content object security, proxying, and caching, was evaluated in detail. Findings indicated that CoAP deployment settings with information-centric properties exhibited similar protocol performances as NDN. Insights revealed that the available CoAP building blocks including CoAP proxies with caches and content object security using OSCORE are nearly complete for building a RESTful information-centric Web of Things (WoT).

Multiparty Communication for the Information-centric WoT. While the NDN architecture supports seamless replication of group content, IP multicast is difficult to implement in low-power wireless regimes due to a challenging layer 2 support. In addition, the synchronous nature of IP multicast endangers successful packet transmissions due to radio interference. Multiparty content dissemination, however, is an important use case in common actuator scenarios (*e.g.,* switching light bulbs) and in particular for over the air software updates. Chapter 11 extended existing CoAP components with few additional functions, and constructed a data-centric WoT that aggregates content requests and replicates responses using CoAP protocol elements. These extended CoAP components provide all ICN-type properties: hop-wise and multi-destination forwarding, on-path caching, request aggregation, response fan-out, and group access to secured content objects. Evaluations of the data-centric WoT model against the NDN and plain CoAP variants indicated that this approach performs similar to NDN, while largely outperforming plain CoAP deployments.

Directions for Future Work

Future research shall concentrate on progressing the data-centric WoT to support the vision of a robust, secure, and interoperable web of constrained, loosely coupled things. Long-term, large-scale deployment studies on real IoT hardware are necessary to learn about insights and optimizations of current design decisions and operational practices. To improve versatility across multiple hardware platforms, ICNLoWPAN needs to be extended to different low-power link technologies, such as Bluetooth Low Energy (BLE). An alignment with adaptation layers for wide-area and cellular technologies is further necessary to enable an efficient content replication over long range (LoRa) communication and NB-IoT. Since device resources in low-cost IoT setups will likely remain scarce, a full exploration of the effects of the proposed coordinative resource actions is necessary. The impact against resource decorrelation when large multiplicities of competing flows are crossing in a densely meshed core network requires further assessment.

Content object security in the CoAP-centered IoT facilitates the data-centric deployment option, but may also introduce an excessive security overhead, notably for packets with a very small payload (*e.g.,* sensor readings), or many sequential fragments of a larger content object (*e.g.,* firmware update). Optimizations for secured content retrievals to ease voluminous data transfers without sacrificing integrity, authenticity, and DoS resistance should be investigated. Further, this deployment option enables the utilization of profound ICN research in CoAP setups. Especially the progressing efforts on in-network caching and on an efficient multiparty distribution of content can be valuable additions to the data-centric WoT without affecting its global interoperability with Internet services.

List of Figures

List of Tables

Bibliography

[1] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," RFC 7228, IETF, May 2014.

[2] IEEE 802.15 Working Group, "IEEE Standard for Low-Rate Wireless Networks," Tech. Rep. IEEE Std 802.15.4™–2015 (Revision of IEEE Std 802.15.4-2011), IEEE, New York, NY, USA, 2016.

[3] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, IETF, September 2007.

[4] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, IETF, June 2014.

[5] A. Banks and R. G. (Eds.), "MQTT Version 3.1.1," oasis standard, OASIS, October 2014.

[6] A. Stanford-Clark and H. L. Truong, "MQTT For Sensor Networks (MQTT-SN) Version 1.2," protocol specification, IBM, November 2013.

[7] W3C, "Web of Things (WoT) Thing Description." `https://www.w3.org/TR/wot-thing-description/`, June 2020. Retrieved 2021-11-17.

[8] M. Koster and C. Bormann, "Semantic Definition Format (SDF) for Data and Interactions of Things," Internet-Draft – work in progress 00, IETF, June 2020.

[9] C. Amsuess, Z. Shelby, M. Koster, C. Bormann, and P. van der Stok, "CoRE Resource Directory," Internet-Draft – work in progress 28, IETF, March 2021.

[10] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," RFC 4838, IETF, April 2007.

[11] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A Survey of Information-Centric Networking," *IEEE Communications Magazine*, vol. 50, pp. 26–36, July 2012.

[12] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[13] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, “Information-Centric networking: Seeing the Forest for the Trees,” in *Proc. of the 10th ACM HotNets Workshop*, HotNets-X, (New York, NY, USA), ACM, 2011.

[14] D. Cheriton and M. Gritter, “TRIAD: A New Next-Generation Internet Architecture.” `http://www-cs-students.stanford.edu/~mgritter/`, July 2000. Retrieved 2021-11-09.

[15] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, “A Data-Oriented (and beyond) Network Architecture,” *SIGCOMM Computer Communications Review*, vol. 37, no. 4, pp. 181–192, 2007.

[16] D. Lagutin, K. Visala, and S. Tarkoma, “Publish/Subscribe for Internet: PSIRP Perspective,” *Future Internet Assembly*, vol. 84, pp. 75–84, 2010.

[17] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, “Developing Information Networking Further: From PSIRP to PURSUIT,” in *Broadband Communications, Networks, and Systems*, (Berlin, Heidelberg), pp. 1–13, Springer Berlin Heidelberg, 2012.

[18] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, “Network of Information (NetInf) - An Information-Centric Networking Architecture,” *Computer Communications*, vol. 36, pp. 721–735, April 2013.

[19] M. Mosko, I. Solis, and C. Wood, “Content-Centric Networking (CCNx) Semantics,” RFC 8569, IETF, July 2019.

[20] M. Mosko, I. Solis, and C. Wood, “Content-Centric Networking (CCNx) Messages in TLV Format,” RFC 8609, IETF, July 2019.

[21] V. Jacobson, D. K. Smetters, J. D. Thornton, and M. F. Plass, “Networking Named Content,” in *5th Int. Conf. on emerging Networking Experiments and Technologies (ACM CoNEXT’09)*, (New York, NY, USA), pp. 1–12, ACM, Dec. 2009.

[22] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, “Named Data Networking,” *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.

[23] G. Carofiglio, L. Muscariello, J. Augé, M. Papalini, M. Sardara, and A. Compagno, “Enabling ICN in the Internet Protocol: Analysis and Evaluation of the Hybrid-ICN Architecture,” in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN ’19, (New York, NY, USA), pp. 55–66, ACM, 2019.

[24] D. Oran, “Considerations in the Development of a QoS Architecture for CCNx-Like Information-Centric Networking Protocols,” RFC 9064, IETF, June 2021.

[25] E. Rescorla and N. Modadugu, “Datagram Transport Layer Security Version 1.2,” RFC 6347, IETF, January 2012.

[26] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, “Object Security for Constrained RESTful Environments (OSCORE),” RFC 8613, IETF, July 2019.

[27] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, “FIT IoT-LAB: A large scale open experimental IoT testbed,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, (Piscataway, NJ, USA), pp. 459–464, IEEE Press, Dec 2015.

[28] E. Baccelli, C. Gündogan, O. Hahm, P. Kietzmann, M. Lenders, H. Petersen, K. Schleiser, T. C. Schmidt, and M. Wählisch, “RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT,” *IEEE Internet of Things Journal*, vol. 5, pp. 4428–4440, December 2018.

[29] M. Lenders, P. Kietzmann, O. Hahm, H. Petersen, C. Gündogan, E. Baccelli, K. Schleiser, T. C. Schmidt, and M. Wählisch, “Connecting the World of Embedded Mobiles: The RIOT Approach to Ubiquitous Networking for the Internet of Things,” Technical Report arXiv:1801.02833, Open Archive: arXiv.org, January 2018.

[30] C. Bormann and P. Hoffman, “Concise Binary Object Representation (CBOR),” RFC 8949, IETF, December 2020.

[31] C. Gündogan, P. Kietzmann, T. C. Schmidt, and M. Wählisch, “Designing a LoWPAN convergence layer for the Information Centric Internet of Things,” *Computer Communications*, vol. 164, pp. 114–123, December 2020.

[32] J. Hui and P. Thubert, “Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks,” RFC 6282, IETF, September 2011.

[33] C. Gündogan, T. C. Schmidt, M. Wählisch, C. Scherb, C. Marxer, and C. Tschudin, “Information-Centric Networking (ICN) Adaptation to Low-Power Wireless Personal Area Networks (LoWPANs),” RFC 9139, RFC Editor, IRTF, November 2021.

[34] R. Braden, D. Clark, and S. Shenker, “Integrated Services in the Internet Architecture: an Overview,” RFC 1633, IETF, June 1994.

[35] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, “Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification,” RFC 2205, IETF, September 1997.

[36] K. Nichols, S. Blake, F. Baker, and D. Black, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,” RFC 2474, IETF, December 1998.

[37] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, IETF, December 1998.

[38] M. Welzl, *Network Congestion Control - Managing Internet Traffic.* Hoboken, NJ,USA: Wiley, 2005.

[39] C. Gündogan, J. Pfender, P. Kietzmann, T. C. Schmidt, and M. Wählisch, "On the Impact of QoS Management in an Information-centric Internet of Things," *Computer Communications*, vol. 154, pp. 160–172, March 2020.

[40] A. Carzaniga, M. Papalini, and A. L. Wolf, "Content-based Publish/Subscribe Networking and Information-centric Networking," in *Proc. of the ACM SIGCOMM WS on Information-centric Networking (ICN '11)*, (New York, NY, USA), pp. 56–61, ACM, 2011.

[41] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. Ramakrishnan, "COPSS: An Efficient Content Oriented Publish/Subscribe System," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'11)*, (Los Alamitos, CA, USA), pp. 99–110, IEEE Computer Society, Oct. 2011.

[42] K. Nichols, "Lessons Learned Building a Secure Network Measurement Framework Using Basic NDN," in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN '19, (New York, NY, USA), pp. 112–122, ACM, 2019.

[43] C. Gündogan, P. Kietzmann, T. C. Schmidt, and M. Wählisch, "A Mobility-compliant Publish Subscribe System for an Information Centric Internet of Things," *Computer Networks*, vol. 203, pp. 1–14, February 2022.

[44] T. C. Schmidt, S. Wölke, N. Berg, and M. Wählisch, "Let's Collect Names: How PANINI Limits FIB Tables in Name Based Routing," in *Proc. of 15th IFIP Networking Conference*, (Piscataway, NJ, USA), pp. 458–466, IEEE Press, May 2016.

[45] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, "Distributed Collaborative Control for Industrial Automation With Wireless Sensor and Actuator Networks," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 12, pp. 4219–4230, 2010.

[46] Q. Wang and J. Jiang, "Comparative Examination on Architecture and Protocol of Industrial Wireless Sensor Network Standards," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2197–2219, 2016.

[47] International Society of Automation, "Wireless Systems for Industrial Automation: Process Control and Related Applications," Tech. Rep. Standard ISA-100.11a-2011, ISA, 2011.

[48] L. M. Feeney, M. Frey, V. Fodor, and M. Günes, “Modes of inter-network interaction in beacon-enabled IEEE 802.15.4 networks,” in *14th Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET*, pp. 1–8, IEEE, June 2015.

[49] S. B. Yaala, F. Théoleyre, and R. Bouallegue, “Cooperative resynchronization to improve the reliability of colocated IEEE 802.15.4 -TSCH networks in dense deployments,” *Ad Hoc Networks*, vol. 64, pp. 112 – 126, 2017.

[50] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, “5G : A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice,” *IEEE Journal on Selected Areas in Communications*, vol. 35, pp. 1201–1221, 06 2017.

[51] A. A. Kumar S., K. Ovsthus, and L. M. Kristensen., “An Industrial Perspective on Wireless Sensor Networks - A Survey of Requirements, Protocols, and Challenges,” *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1391–1412, 2014.

[52] N. Sastry and D. Wagner, “Security Considerations for IEEE 802.15.4 Networks,” in *Proc. of the 3rd ACM Workshop on Wireless Security (WiSe ’04)*, (New York, NY, USA), pp. 32–42, ACM, 2004.

[53] S. M. Sajjad and M. Yousaf, “Security analysis of IEEE 802.15.4 MAC in the context of Internet of Things (IoT),” in *Conference on Information Assurance and Cyber Security (CIACS ’14)*, pp. 9–14, IEEE, 2014.

[54] T. Watteyne, M. Palattella, and L. Grieco, “Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement,” RFC 7554, IETF, May 2015.

[55] X. Vilajosana, K. Pister, and T. Watteyne, “Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration,” RFC 8180, IETF, May 2017.

[56] L. M. Feeney and P. Gunningberg, “Avoiding an IoT ’Tragedy of the Commons’,” in *Proc. of the 16th International Conference on Mobile Systems, Applications, and Services (MobiSys ’18)*, (New York, NY, USA), pp. 495–497, ACM, 2018.

[57] C. Bockelmann, N. Pratas, H. Nikopour, K. Au, T. Svensson, C. Stefanovic, P. Popovski, and A. Dekorsy, “Massive machine-type communications in 5g: physical and MAC-layer solutions,” *IEEE Communications Magazine*, vol. 54, no. 9, pp. 59–65, 2016.

[58] Y. Cai, Z. Qin, F. Cui, G. Y. Li, and J. A. McCann, “Modulation and Multiple Access for 5G Networks,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 629–646, 2018.

[59] K. Hartke, "Observing Resources in the Constrained Application Protocol (CoAP)," RFC 7641, IETF, September 2015.

[60] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, IETF, August 2018.

[61] M. Zhang, V. Lehman, and L. Wang, "Scalable name-based data synchronization for named data networking," in *IEEE INFOCOM'17*, INFOCOM '17, (Los Alamitos, CA, USA), pp. 1–9, IEEE Computer Society, 2017.

[62] C. Gündogan, P. Kietzmann, T. C. Schmidt, and M. Wählisch, "HoPP: Robust and Resilient Publish-Subscribe for an Information-Centric Internet of Things," in *Proc. of the 43rd IEEE Conference on Local Computer Networks (LCN)*, (Piscataway, NJ, USA), pp. 331–334, IEEE Press, Oct. 2018.

[63] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," *ACM Trans. Comput. Syst.*, vol. 2, pp. 277–288, Nov 1984.

[64] X. de Carnavalet and M. Mannan, "Killed by Proxy: Analyzing Client-end TLS Interception Software," in *Network and Distributed System Security Symposium (NDSS)*, Internet Society, 2016.

[65] R. Holz, T. Riedmaier, N. Kammenhuber, and G. Carle, "X.509 Forensics: Detecting and Localising the SSL/TLS Men-in-the-Middle," in *Computer Security – ESORICS 2012*, (Berlin, Heidelberg), pp. 217–234, Springer Berlin Heidelberg, 2012.

[66] C. Gündogan, C. Amsüss, T. C. Schmidt, and M. Wählisch, "IoT Content Object Security with OSCORE and NDN: A First Experimental Comparison," in *Proc. of 19th IFIP Networking Conference*, (Piscataway, NJ, USA), pp. 19–27, IEEE Press, June 2020.

[67] C. Bormann, "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," RFC 7400, IETF, November 2014.

[68] C. Gündogan, T. C. Schmidt, M. Wählisch, C. Scherb, C. Marxer, and C. Tschudin, "ICN Adaptation to LowPAN Networks (ICN LoWPAN)," IRTF Internet Draft – work in progress 10, IRTF, February 2021.

[69] Atmel, *Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE, SP100, WirelessHART, and ISM Applications.* Atmel Corporation, September 2009.

[70] G. Hansch, P. Schneider, and G. S. Brost, "Deriving Impact-Driven Security Requirements and Monitoring Measures for Industrial IoT," in *Proc. of the 5th on Cyber-Physical System Security Workshop (CPSS'19)*, (New York), pp. 37–45, ACM, 2019.

[71] G. Bernieri, M. Conti, and G. Pozzan, “AMON: An Automaton MONitor for Industrial Cyber-Physical Security,” in *Proc. of the 14th International Conference on Availability, Reliability and Security (ARES ’19)*, (New York), pp. 1–10, ACM, 2019.

[72] Modbus–IDA, “Modbus application protocol specification v1. 1b,” tech. rep., Modbus–IDA, 2006.

[73] M. Nolan, M. J. McGrath, M. Spoczynski, and D. Healy, “Adaptive Industrial IOT/CPS Messaging Strategies for Improved Edge Compute Utility,” in *Proc. of the Workshop on Fog Computing and the IoT (IoT-Fog ’19)*, (New York), pp. 16–20, ACM, 2019.

[74] B. Chun, B. Oh, C. Cho, and D. Lee, “Design and Implementation of Lightweight Messaging Middleware for Edge Computing,” in *Proceedings of the 6th International Conference on Control, Mechatronics and Automation (ICCMA ’18)*, (New York), pp. 170–174, ACM, 2018.

[75] L. Eggert, “Towards Securing the Internet of Things with QUIC,” in *Proc. of 3rd NDSS Workshop on Decentralized IoT Systems and Security (DISS)*, Internet Society (ISOC), 2020.

[76] J. Iyengar and M. Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport,” Internet-Draft – work in progress 34, IETF, January 2021.

[77] Q. D. Coninck and O. Bonaventure, “Multipath Extensions for QUIC (MP-QUIC),” Internet-Draft – work in progress 07, IETF, May 2021.

[78] I. Swett, M.-J. Montpetit, V. Roca, and F. Michel, “Coding for QUIC,” Internet-Draft – work in progress 04, IETF, March 2020.

[79] Q. D. Coninck and O. Bonaventure, “Multipath QUIC: Design and Evaluation,” in *Proc. of CoNEXT ’17*, (New York, NY, USA), pp. 160–166, ACM, Dec. 2017.

[80] F. Michel, Q. D. Coninck, and O. Bonaventure, “QUIC-FEC: Bringing the benefits of Forward Erasure Correction to QUIC,” in *Proc. of 19th IFIP Networking Conference*, (Piscataway, NJ, USA), pp. 1–9, IEEE Press, May 2019.

[81] M. Iglesias-Urkia, A. Orive, and A. Urbieta, “Analysis of CoAP Implementations for Industrial Internet of Things: A Survey,” *Procedia Computer Science*, vol. 109, pp. 188–195, 2017.

[82] J. Dizdarevic, F. Carpio, A. Jukan, and X. Masip-Bruin, “Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration,” *ACM Comput. Surv.*, vol. 51, pp. 116–1 – 116–29, Jan. 2019.

[83] D. Koutras, G. Stergiopoulos, T. Dasaklis, P. Kotzanikolaou, D. Glynos, and C. Douligeris, “Security in IoMT Communications: A Survey,” *Sensors*, vol. 20, no. 17, p. 4828, 2020.

[84] N. F. Syed, Z. Baig, A. Ibrahim, and C. Valli, “Denial of service attack detection through machine learning for the IoT,” *Journal of Information and Telecommunication*, vol. 4, no. 4, pp. 482–503, 2020.

[85] C. Lerche, K. Hartke, and M. Kovatsch, “Industry adoption of the Internet of Things: A constrained application protocol survey,” in *Proc. 17th IEEE International Conf on Emerging Technologies & Factory Automation (ETFA)*, (Piscataway, NJ, USA), pp. 1–6, IEEE, 2012.

[86] 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 *Proc. of 6th International Conf on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, (Washington, DC, USA), pp. 702–707, IEEE Computer Society, 2012.

[87] A. Ludovici, P. Moreno, and A. Calveras, “TinyCoAP: A Novel Constrained Application Protocol (CoAP) Implementation for Embedding RESTful Web Services in Wireless Sensor Networks Based on TinyOS,” *J. Sensor and Actuator Networks*, vol. 2, no. 2, pp. 288–315, 2013.

[88] C. P. Kruger and G. P. Hancke, “Benchmarking Internet of things devices,” in *Proc. of 12th IEEE International Conf on Industrial Informatics (INDIN)*, (Piscataway, NJ, USA), pp. 611–616, IEEE, 2014.

[89] A. Elmangoush, R. Steinke, T. Magedanz, A. A. Corici, A. Bourreau, and A. Al-Hezmi, “Application-derived communication protocol selection in M2M platforms for smart cities,” in *Proc. of 18th International Conference on Intelligence in Next Generation Networks (ICIN)*, (Piscataway, NJ, USA), pp. 76–82, IEEE, 2015.

[90] D. Mishra, R. S. Yadav, K. K. Agrawal, and A. Abbas, “Study of Application Layer Protocol for Real-Time Monitoring and Maneuvering,” in *International Conference on Innovative Computing and Communications* (A. Khanna, D. Gupta, S. Bhattacharyya, V. Snasel, J. Platos, and A. E. Hassanien, eds.), (Singapore), pp. 439–449, Springer Singapore, 2020.

[91] J. J. R. Rodríguez, J. F. C. García, and E. J. A. üello Prada, “Toward Automatic and Remote Monitoring of the Pain Experience: An Internet of Things (IoT) Approach,” in *Applied Technologies* (M. Botto-Tobar, M. Z. Vizuete, P. Torres-Carrión, S. M. León, G. P. Vásquez, and B. Durakovic, eds.), (Cham), pp. 194–206, Springer International Publishing, 2020.

[92] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in *Proc. of ISSNIP*, (Piscataway, NJ, USA), pp. 1–6, IEEE, 2014.

[93] Y. Chen and T. Kunz, "Performance evaluation of IoT protocols under a constrained wireless access network," in *International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, (Piscataway, NJ, USA), pp. 1–7, IEEE, 2016.

[94] M. Martí, C. Garcia-Rubio, and C. Campo, "Performance Evaluation of CoAP and MQTT-SN in an IoT Environment," in *13th Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI'19)*, vol. 31, p. 49, 2019.

[95] D. P. Proos and N. Carlsson, "Performance Comparison of Messaging Protocols and Serialization Formats for Digital Twins in IoV," in *Proc. of 19th IFIP Networking Conference*, (Piscataway, NJ, USA), pp. 10–18, IEEE Press, June 2020.

[96] A. Larmo, A. Ratilainen, and J. Saarinen, "Impact of CoAP and MQTT on NB-IoT system performance," *Sensors*, vol. 19, p. 7, 2018.

[97] W. Shang, Y. Yu, T. Liang, B. Zhang, , and L. Zhang, "NDN-ACE: Access Control for Constrained Environments over Named Data Networking," Technical Report NDN-0036, NDN, December 2015.

[98] B. Mathieu, C. Westphal, and P. Truong, "Towards the usage of ccn for iot networks," in *Internet of Things (IoT) in 5G Mobile Technologies*, pp. 3–24, Cham, Switzerland: Springer, 2016.

[99] E. M. Schooler, D. Zage, J. Sedayao, H. Moustafa, A. Brown, and M. Ambrosin, "An Architectural Vision for a Data-Centric IoT: Rethinking Things, Trust and Clouds," in *IEEE 37th Intern. Conference on Distributed Computing Systems (ICDCS)*, (Piscataway, NJ, USA), pp. 1717–1728, IEEE, June 2017.

[100] H. M. A. Islam, D. Lagutin, A. Ylä-Jääski, N. Fotiou, and A. V. Gurtov, "Transparent CoAP Services to IoT Endpoints through ICN Operator Networks," *Sensors*, vol. 19, no. 6, p. 1339, 2019.

[101] A. L. R. Madureira, F. R. C. Araújo, G. B. Araújo, and L. N. Sampaio, "NDN Fabric: Where the Software-Defined Networking Meets the Content-Centric Model," *IEEE Transactions on Network and Service Management*, 2020.

[102] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, "Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control and NDN," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, (Piscataway, NJ, USA), pp. 394–398, IEEE, 2013.

[103] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Information Centric Networking in IoT scenarios: The case of a smart home," in *Proc. of IEEE International Conference on Communications (ICC)*, (Piscataway, NJ, USA), pp. 648–653, IEEE, June 2015.

[104] M. Frey, C. Gündogan, P. Kietzmann, M. Lenders, H. Petersen, T. C. Schmidt, F. Shzu-Juraschek, and M. Wählisch, "Security for the Industrial IoT: The Case for Information-Centric Networking," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT) (WF-IoT 2019)*, (Piscataway, NJ, USA), pp. 424–429, IEEE Press, April 2019.

[105] C. Gündogan, C. Amsüss, T. C. Schmidt, and M. Wählisch, "Toward a RESTful Information-Centric Web of Things: A Deeper Look at Data Orientation in CoAP," in *Proc. of 7th ACM Conference on Information-Centric Networking (ICN)*, (New York), pp. 77–88, ACM, September 2020.

[106] C. Tschudin, C. Scherb, *et al.*, "CCN Lite: Lightweight implementation of the Content Centric Networking protocol," 2018.

[107] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors.," in *Proc. of IEEE Local Computer Networks (LCN)*, (Los Alamitos, CA, USA), pp. 455–462, IEEE Computer Society, 2004.

[108] W. Shang, A. Afanasyev, and L. Zhang, "The Design and Implementation of the NDN Protocol Stack for RIOT-OS," in *Proc. of IEEE GLOBECOM 2016*, (Washington, DC, USA), pp. 1–6, IEEE, 2016.

[109] P. Kietzmann, C. Gündogan, T. C. Schmidt, O. Hahm, and M. Wählisch, "The Need for a Name to MAC Address Mapping in NDN: Towards Quantifying the Resource Gain," in *Proc. of 4th ACM Conference on Information-Centric Networking (ICN)*, (New York, NY, USA), pp. 36–42, ACM, September 2017.

[110] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information Centric Networking in the IoT: Experiments with NDN in the Wild," in *Proc. of 1st ACM Conf. on Information-Centric Networking (ICN-2014)*, (New York), pp. 77–86, ACM, September 2014.

[111] C. Gündogan, P. Kietzmann, M. Lenders, H. Petersen, T. C. Schmidt, and M. Wählisch, "NDN, CoAP, and MQTT: A Comparative Measurement Study in the IoT," in *Proc. of 5th ACM Conference on Information-Centric Networking (ICN)*, (New York, NY, USA), pp. 159–171, ACM, September 2018.

[112] M. Gritter and D. R. Cheriton, "An Architecture for Content Routing Support in the Internet," in *Proc. USITS'01*, (Berkeley, CA, USA), pp. 4–4, USENIX Association, 2001.

[113] D. Trossen, M. J. Reed, J. Riihijärvi, M. Georgiades, N. Fotiou, and G. Xylomenos, "IP over ICN - The better IP?," in *2015 European Conference on Networks and Communications (EuCNC)*, (Los Alamitos, CA, USA), pp. 413–417, IEEE Computer Society, June 2015.

[114] M. Mosko and C. Tschudin, "ICN 'Begin-End' Hop by Hop Fragmentation," Internet-Draft – work in progress 02, IETF, December 2016.

[115] J. Shi and B. Zhang, "NDNLP: A Link Protocol for NDN," NDN, Technical Report NDN-0006, NDN Team, July 2012.

[116] C. Ghali, A. Narayanan, D. Oran, G. Tsudik, and C. A. Wood, "Secure Fragmentation for Content-Centric Networks," in *Proc. of the 14th International Symposium on Network Computing and Applications*, (Piscataway, NJ, USA), pp. 47–56, IEEE Press, 2015.

[117] M. S. Lenders, T. C. Schmidt, and M. Wählisch, "A Lesson in Scaling 6LoWPAN – Minimal Fragment Forwarding in Lossy Networks," in *Proc. of the 44rd IEEE Conference on Local Computer Networks (LCN)*, (Piscataway, NJ, USA), pp. 438–446, IEEE Press, Oct. 2019.

[118] Y. Yang and T. Song, "Local Name Translation for Succinct Communication Towards Named Data Networking of Things," *IEEE Communications Letters*, vol. 22, pp. 2551–2554, Dec. 2018.

[119] P. Thubert and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch," RFC 8025, IETF, November 2016.

[120] T. Watteyne, P. Thubert, and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multihop IPv6 Network," Internet-Draft – work in progress 15, IETF, March 2020.

[121] P. Thubert, "6LoWPAN Selective Fragment Recovery," Internet-Draft – work in progress 21, IETF, March 2020.

[122] M. S. Lenders, T. C. Schmidt, and M. Wählisch, "Fragment Forwarding in Lossy Networks," *IEEE Access*, vol. 9, pp. 143969 – 143987, October 2021.

[123] M. S. Lenders, C. Gündogan, T. C. Schmidt, and M. Wählisch, "Connecting the Dots: Selective Fragment Recovery in ICNLoWPAN," in *Proc. of 7th ACM Conference on Information-Centric Networking (ICN)*, (New York), pp. 70–76, ACM, September 2020.

[124] C. Gündogan, T. C. Schmidt, D. Oran, and M. Wählisch, "Alternative Delta Time Encoding for CCNx Using Compact Floating-Point Arithmetic," IRTF Internet Draft – work in progress 00, IRTF, April 2022.

[125] IEEE 754 Working Group, "IEEE Standard for Floating-Point Arithmetic," Tech. Rep. IEEE Std 754-2019 (Revision of IEEE Std 754-2008), IEEE, New York, NY, USA, 2019.

[126] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges," RFC 7927, IETF, July 2016.

[127] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, "Backscatter from the Data Plane – Threats to Stability and Security in Information-Centric Network Infrastructure," *Computer Networks*, vol. 57, pp. 3192–3206, Nov. 2013.

[128] D. Oran, "Considerations in the development of a QoS Architecture for CCNx-like ICN protocols," Internet-Draft – work in progress 06, IETF, November 2020.

[129] D. Oran, "Maintaining CCNx or NDN flow balance with highly variable data object sizes," Internet-Draft – work in progress 07, IETF, May 2022.

[130] I. Moiseenko and D. Oran, "Flow Classification in Information Centric Networking," Internet-Draft – work in progress 07, IETF, January 2021.

[131] A. Jangam, P. suthar, and M. Stolic, "QoS Treatments in ICN using Disaggregated Name Components," Internet-Draft – work in progress 02, IETF, March 2020.

[132] C. Gündogan, T. C. Schmidt, M. Wählisch, M. Frey, F. Shzu-Juraschek, and J. Pfender, "Quality of Service for ICN in the IoT," IRTF Internet Draft – work in progress 01, IRTF, July 2019.

[133] C. Tsilopoulos and G. Xylomenos, "Supporting Diverse Traffic Types in Information Centric Networks," in *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '11, (New York, NY, USA), pp. 13–18, ACM, 2011.

[134] M. Mahdian, S. Arianfar, J. Gibson, and D. Oran, "MIRCC: Multipath-aware ICN Rate-based Congestion Control," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, (New York, NY, USA), pp. 1–10, ACM, 2016.

[135] N. Dukkipati, *Rate control protocol (rcp): Congestion control to make flows complete quickly.* PhD thesis, Stanford University, Stanford, CA, USA, 2008.

[136] M. F. Al-Naday, A. Bontozoglou, V. G. Vassilakis, and M. J. Reed, "Quality of Service in an Information-Centric Network," in *Proceedings of Globecom 2014*, pp. 1861–1866, 2014.

[137] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1473–1499, 2015.

[138] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in Named Data Networking for the wireless Internet of Things," in *2015 International Conference on Recent Advances in Internet of Things (RIoT)*, pp. 1–6, IEEE, 2015.

[139] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic In-Network Caching for Information-Centric Networks," in *Proc. of the 2nd edition of the ICN workshop on Information-centric networking*, ICN '12, (New York, NY, USA), pp. 55–60, ACM, 2012.

[140] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, and Y. Ji, "Performance of probabilistic caching and cache replacement policies for Content-Centric Networks," in *39th Annual IEEE Conference on Local Computer Networks*, (Piscataway, NJ, USA), pp. 99–106, IEEE, 2014.

[141] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "On the Performance of Caching and Forwarding in Information-Centric Networking for the IoT," in *Proceedings of the International Conference on Wired/Wireless Internet Communication (WWIC)*, (Cham, Switzerland), pp. 313–326, Springer, 2015.

[142] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Information-Centric Networking Based Caching and Naming Schemes for Internet of Things: A Survey and Future Research Directions," *arXiv preprint arXiv:1710.03473*, 2017.

[143] J. Pfender, A. Valera, and W. K. G. Seah, "Performance Comparison of Caching Strategies for Information-Centric IoT," in *Proc. of the 5th ACM Conference on Information-Centric Networking*, ICN '18, (New York, NY, USA), pp. 43–53, ACM, 2018.

[144] D. D. Van and Q. Ai, "An efficient in-network caching decision algorithm for Internet of things," *International Journal of Communication Systems*, vol. 31, no. 8, p. e3521, 2018.

[145] B. Chen, L. Liu, Z. Zhang, W. Yang, and H. Ma, "BRR-CVR: A Collaborative Caching Strategy for Information-Centric Wireless Sensor Networks," in *2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, (Piscataway, NJ, USA), pp. 31–38, IEEE, 2016.

[146] L. Zhang, J. Zhao, and Z. Shi, "LF: A Caching Strategy for Named Data Mobile Ad Hoc Networks," in *Proc. of the 4th International Conference on Computer Engineering and Networks*, (Cham, Switzerland), pp. 279–290, Springer, 2015.

[147] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "Less for More" in Information-Centric Networks (Extended Version)," in *Special Issue on Information-Centric Networking*, vol. 36, pp. 758–770, Elsevier, 2013.

[148] S. Naz, R. Naveed Bin Rais, P. A. Shah, S. Yasmin, A. Qayyum, S. Rho, and Y. Nam, "A dynamic caching strategy for CCN-based MANETs," *Computer Networks*, vol. 142, pp. 93–107, 2018.

[149] L. Zhou, T. Zhang, X. Xu, Z. Zeng, and Y. Liu, “Broadcasting based neighborhood cooperative caching for content centric ad hoc networks,” in *2015 IEEE/CIC International Conference on Communications in China (ICCC)*, (Piscataway, NJ, USA), pp. 1–5, IEEE, 2015.

[150] Y. Sun, T. Zhang, R. Wang, W. Feng, and P. Chen, “Topology Potential Based Probability Caching Strategy for Content Centric Ad Hoc Networks,” *Journal of Residuals Science & Technology*, vol. 13, no. 6, 2016.

[151] S. Ioannidis and E. Yeh, “Jointly Optimal Routing and Caching for Arbitrary Network Topologies,” in *4th ACM Conference on Information-Centric Networking*, ACM-ICN ’17, (New York, NY, USA), pp. 77–87, ACM, 2017.

[152] J. Pfender, A. Valera, and W. K. Seah, “Easy as ABC: A Lightweight Centrality-Based Caching Strategy for Information-Centric IoT,” in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN ’19, (New York, NY, USA), pp. 100–111, ACM, 2019.

[153] Y. Liu, D. Zhu, and W. Ma, “A novel cooperative caching scheme for Content Centric Mobile Ad Hoc Networks,” in *2016 IEEE Symposium on Computers and Communication (ISCC)*, (Piscataway, NJ, USA), pp. 824–829, IEEE, 2016.

[154] Z. Li and G. Simon, “Time-Shifted TV in Content Centric Networks: The Case for Cooperative In-Network Caching,” in *2011 IEEE International Conference on Communications (ICC)*, (Piscataway, NJ, USA), pp. 1–6, IEEE, 2011.

[155] Y. Zeng and X. Hong, “A caching strategy in mobile ad hoc named data network,” in *2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)*, pp. 805–809, 2011.

[156] O. Hahm, E. Baccelli, T. C. Schmidt, M. Wählisch, C. Adjih, and L. Massoulié, “Low-power Internet of Things with NDN and Cooperative Caching,” in *Proc. of 4th ACM Conference on Information-Centric Networking (ICN)*, (New York, NY, USA), pp. 98–108, ACM, September 2017.

[157] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, “RIOT OS: Towards an OS for the Internet of Things,” in *Proc. of the 32nd IEEE INFOCOM. Poster*, (Piscataway, NJ, USA), pp. 79–80, IEEE Press, 2013.

[158] J. Seedorf, M. Arumaithurai, A. Tagami, K. Ramakrishnan, and N. Blefari-Melazzi, “Research Directions for Using ICN in Disaster Scenarios,” Internet-Draft – work in progress 07, IETF, June 2019.

[159] X. Jiang, J. Taneja, J. Ortiz, A. Tavakoli, P. Dutta, J. Jeong, D. Culler, P. Levis, and S. Shenker, "An Architecture for Energy Management in Wireless Sensor Networks," *SIGBED Review*, vol. 4, pp. 31–36, July 2007.

[160] J. Vasseur, "Terms Used in Routing for Low-Power and Lossy Networks," RFC 7102, IETF, January 2014.

[161] J. Hester and J. Sorber, "The Future of Sensing is Batteryless, Intermittent, and Awesome," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, (New York, NY, USA), pp. 1–6, Association for Computing Machinery, 2017.

[162] P. Ginzboorg, T. Kärkkäinen, A. Ruotsalainen, M. Andersson, and J. Ott, "DTN Communication in a Mine," in *2nd Extreme Workshop on Communications*, ACM, Sept. 2010.

[163] C. Gündogan, P. Kietzmann, T. C. Schmidt, M. Lenders, H. Petersen, M. Wählisch, M. Frey, and F. Shzu-Juraschek, "Information-Centric Networking for the Industrial IoT," in *Proc. of 4th ACM Conference on Information-Centric Networking (ICN), Demo Session*, (New York, NY, USA), pp. 214–215, ACM, September 2017.

[164] G. Tyson, N. Sastry, R. Cuevas, I. Rimac, and A. Mauthe, "A Survey of Mobility in Information-centric Networks," *Commun. ACM*, vol. 56, pp. 90–98, Dec. 2013.

[165] Y. Zhang, Z. Xia, S. Mastorakis, and L. Zhang, "KITE: Producer Mobility Support in Named Data Networking," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, (New York, NY, USA), pp. 125–136, ACM, September 2018.

[166] Y. Zhang, A. Afanasyev, J. Burke, and L. Zhang, "A survey of mobility support in Named Data Networking," in *Proc. of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, (Piscataway, NJ, USA), pp. 83–88, IEEE, 2016.

[167] J. Augé, G. Carofiglio, G. Grassi, L. Muscariello, G. Pau, and X. Zeng, "MAP-Me: Managing Anchor-Less Producer Mobility in Content-Centric Networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 596–610, 2018.

[168] D. Hernandez, L. Gameiro, C. Senna, M. Luís, and S. Sargento, "Handling Producer and Consumer Mobility in IoT Publish-Subscribe Named Data Networks," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 868–884, 2021.

[169] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "MobilityFirst: a robust and trustworthy mobility-centric architecture for the future internet," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 16, pp. 2–13, December 2012.

[170] F. Hermans, E. Ngai, and P. Gunningberg, "Global source mobility in the content-centric networking architecture," in *Proceedings of the 1st ACM Workshop on Emerging Name-*

Oriented Mobile Networking Design - Architecture, Algorithms, and Applications, NoM '12, (New York, NY, USA), pp. 13–18, ACM, 2012.

[171] X. Jiang, J. Bi, Y. Wang, P. Lin, and Z. Li, "A content provider mobility solution of named data networking," in *Proc. of the 20th IEEE International Conference on Network Protocols (ICNP 2013)*, pp. 1–2, 2012.

[172] A. Azgin, R. Ravindran, A. Chakraborti, and G. Wang, "Seamless Producer Mobility as a Service in Information Centric Networks," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ICN '16, (New York, NY, USA), pp. 243–248, ACM, 2016.

[173] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang, "VANET via Named Data Networking," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 410–415, IEEE, 2014.

[174] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and S. Gannouni, "AFIRM: Adaptive forwarding based link recovery for mobility support in NDN/IoT networks," *Future Generation Computer Systems*, vol. 87, pp. 351–363, 2018.

[175] V. Sivaraman, D. Guha, and B. Sikdar, "Towards Seamless Producer Mobility in Information Centric Vehicular Networks," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5, IEEE, 2020.

[176] H. M. Islam, A. Lukyanenko, S. Tarkoma, and A. Yla-Jaaski, "Towards disruption tolerant ICN," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, pp. 212–219, IEEE, 2015.

[177] S. Arabi, E. Sabir, and H. Elbiaze, "Information-centric networking meets delay tolerant networking: Beyond edge caching," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2018.

[178] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A Case for Stateful Forwarding Plane," Tech. Rep. NDN-0002, PARC, July 2012.

[179] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the Role of Routing in Named Data Networking," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, (New York, NY, USA), pp. 27–36, ACM, 2014.

[180] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, and J. Crowcroft, "Pro-Diluvian: Understanding Scoped-Flooding for Content Discovery in Information-Centric Networking," in *2Nd ACM Conference on Information-Centric Networking*, ACM-ICN '15, (New York, NY, USA), pp. 9–18, ACM, 2015.

[181] J. J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, “A light-weight forwarding plane for content-centric networks,” in *2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–7, IEEE, Feb 2016.

[182] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, “Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised),” RFC 4601, IETF, August 2006.

[183] W. Shang, A. Gawande, M. Zhang, A. Afanasyev, J. Burke, L. Wang, and L. Zhang, “Publish-subscribe communication in building management systems over named data networking,” in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–10, IEEE, 2019.

[184] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. A. Siris, and G. C. Polyzos, “Caching and mobility support in a publish-subscribe internet architecture,” *IEEE Communications Magazine*, vol. 50, no. 7, pp. 52–58, 2012.

[185] A. Detti, D. Tassetto, N. B. Melazzi, and F. Fedi, “Exploiting content centric networking to develop topic-based, publish-subscribe MANET systems,” *Ad Hoc Networks*, vol. 24, pp. 115–133, 2015.

[186] U. Amozarrain and M. Larrea, “Full Mobility and Fault Tolerance in Content-Based Publish/Subscribe,” tech. rep., University of the Basque Country UPV/EHU, 2019.

[187] T. Yu, Z. Zhang, X. Ma, P. Moll, and L. Zhang, “A Pub/Sub API for NDN-Lite with Built-in Security,” Technical Report NDN-0071, NDN, Jan. 2021.

[188] J. Lee, S. M. Hwang, T. Abdelzaher, K. Marcus, and K. Chan, “Pub/sub-sum: A content summarization pub/sub protocol for information-centric networks,” in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, pp. 847–852, Nov. 2019.

[189] H. Jung, K. Choi, H. Kim, and S. Kim, “A Networking Scheme for Large-Scale Pub/Sub Service over NDN,” in *International Conference on Information and Communication Technology Convergence (ICTC ’19)*, pp. 1195–1200, IEEE, 2019.

[190] S. Y. Oh, D. Lau, and M. Gerla, “Content Centric Networking in tactical and emergency MANETs,” in *2010 IFIP Wireless Days*, (Piscataway, NJ, USA), pp. 1–5, IEEE, Oct 2010.

[191] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, “Bulk of Interest: Performance Measurement of Content-Centric Routing,” in *Proc. of ACM SIGCOMM, Poster Session*, (New York), pp. 99–100, ACM, August 2012.

[192] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, “DoS and DDoS in Named Data Networking,” in *Proc. of ICCCN*, pp. 1–7, IEEE, 2013.

[193] S. Al-Sheikh, M. Wählisch, and T. C. Schmidt, "Revisiting Countermeasures Against NDN Interest Flooding," in *2nd ACM Conference on Information-Centric Networking, Poster Session*, ICN 2015, (New York), pp. 195–196, ACM, Oct. 2015.

[194] G. C. Polyzos and N. Fotiou, "Building a reliable Internet of Things using Information-Centric Networking," *Journal of Reliable Intelligent Environments*, vol. 1, no. 1, pp. 47–58, 2015.

[195] J. J. Garcia-Luna-Aceves, "ADN: An Information-Centric Networking Architecture for the Internet of Things," in *Proc. of the 2nd International Conference on Internet-of-Things Design and Implementation*, IoTDI '17, (New York, NY, USA), pp. 27–36, ACM, 2017.

[196] D. Saxena, V. Raychoudhury, and N. SriMahathi, "SmartHealth-NDNoT: Named Data Network of Things for Healthcare Services," in *Proc. of Workshop on Pervasive Wireless Healthcare (MobileHealth)*, (New York, NY, USA), pp. 45–50, ACM, 2015.

[197] O. Hahm, C. Adjih, E. Baccelli, T. C. Schmidt, and M. Wählisch, "ICN over TSCH: Potentials for Link-Layer Adaptation in the IoT," in *Proc. of 3rd ACM Conf. on Information-Centric Networking (ICN 2016), Poster Session*, (New York, NY, USA), pp. 195—196, ACM, September 2016.

[198] B. Ahlgren, A. Lindgren, and Y. Wu, "Demo: Experimental Feasibility Study of CCN-lite on Contiki Motes for IoT Data Streams," in *Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking*, (New York, NY, USA), pp. 221–222, ACM, 2016.

[199] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Named data networking for IoT: An architectural perspective," in *2014 European Conference on Networks and Communications (EuCNC)*, (Piscataway, NJ, USA), pp. 1–5, IEEE, June 2014.

[200] R. Ravindran, A. Chakraborti, S. Amin, and J. Chen, "Support for Notifications in CCN," Internet-Draft – work in progress 01, IETF, July 2017.

[201] C. Gündogan, P. Kietzmann, T. C. Schmidt, M. Lenders, H. Petersen, M. Wählisch, M. Frey, and F. Shzu-Juraschek, "Demo: Seamless Producer Mobility for the Industrial Information-Centric Internet," in *Proc. of 16th ACM International Conference on Mobile Systems, Applications (MobiSys), Demo Session*, (New York, NY, USA), ACM, June 2018.

[202] 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, IETF, March 2012.

[203] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," RFC 6206, IETF, March 2011.

[204] O. Gnawali and P. Levis, “The Minimum Rank with Hysteresis Objective Function,” RFC 6719, IETF, September 2012.

[205] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann, “Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs),” RFC 6775, IETF, November 2012.

[206] C. Tschudin, C. Wood, M. Mosko, and D. Oran, “File-Like ICN Collections (FLIC),” Internet-Draft – work in progress 02, IETF, November 2019.

[207] C. Perkins, D. Johnson, and J. Arkko, “Mobility Support in IPv6,” RFC 6275, IETF, July 2011.

[208] M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, “NLSR: Named-data Link State Routing Protocol,” in *3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN ’13, (New York, NY, USA), pp. 15–20, ACM, 2013.

[209] T. C. Schmidt and M. Wählisch, “Predictive versus Reactive – Analysis of Handover Performance and Its Implications on IPv6 and Multicast Mobility,” *Telecommunication Systems*, vol. 30, pp. 123–142, November 2005.

[210] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6),” RFC 4861, IETF, September 2007.

[211] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed diffusion for wireless sensor networking,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[212] Y. Yu, A. Afanasyev, J. Seedorf, Z. Zhang, and L. Zhang, “NDN DeLorean: an authentication system for data archives in named data networking,” in *4th ACM Conference on Information-Centric Networking*, ACM-ICN ’17, (New York, NY, USA), pp. 11–21, ACM, 2017.

[213] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, “A Firmware Update Architecture for Internet of Things,” RFC 9019, IETF, April 2021.

[214] B. Moran, H. Tschofenig, and H. Birkholz, “An Information Model for Firmware Updates in IoT Devices,” Internet-Draft – work in progress 08, IETF, October 2020.

[215] B. Moran, H. Tschofenig, H. Birkholz, and K. Zandberg, “A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest,” Internet-Draft – work in progress 11, IETF, December 2020.

[216] ZigBee Alliance, “ZigBee Specification,” Specification Document 05-3474-21, ZigBee Alliance, August 2015.

[217] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," in *IEEE Symposium on Security and Privacy (SP)*, (Piscataway, NJ, USA), pp. 195–212, IEEE Press, 2017.

[218] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," *Wireless Networks*, vol. 8, pp. 169–185, 2002.

[219] J. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. of the 2nd Int. Conf. on Embedded Networked Sensor Systems (SenSys'04)*, (New York, NY, USA), pp. 81–94, ACM, 2004.

[220] T. Stathopoulos, J. Heidemann, and D. Estrin, "A remote code update mechanism for wireless sensor networks," Tech. Rep. 30, Center for Embedded Networked Sensing (CENS), Los Angeles, CA, USA, 2003.

[221] C. Bormann and Z. Shelby, "Block-Wise Transfers in the Constrained Application Protocol (CoAP)," RFC 7959, IETF, August 2016.

[222] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent Advances in Information-Centric Networking-Based Internet of Things (ICN-IoT)," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2128–2158, 2019.

[223] M. Mosko and C. Tschudin, "ICN 'Begin-End' Hop by Hop Fragmentation," Internet-Draft – work in progress 02, IETF, December 2016.

[224] P. Gusev and J. Burke, "NDN-RTC: Real-Time Videoconferencing over Named Data Networking," in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, ICN '15, (New York, NY, USA), pp. 117–126, ACM, 2015.

[225] M. Mosko, "CCNx Content Object Chunking," Internet-Draft – work in progress 02, IETF, June 2016.

[226] Z. Zhu and A. Afanasyev, "Let's ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking," in *Proc. of the 21st IEEE International Conference on Network Protocols (ICNP 2013)*, (Piscataway, NJ, USA), pp. 1–10, IEEE, 2013.

[227] M. Zhang, V. Lehman, and L. Wang, "PartialSync: Efficient Synchronization of a Partial Namespace in NDN," Tech. Rep. NDN-0039-1, NDN, June 2016.

[228] C. Percival, "Naive differences of executable code," technical report, daemonology.net, 2003.

[229] J. Koshy and R. Pandey, "Remote incremental linking for energy-efficient reprogramming of sensor networks," in *Proceeedings of the 2nd European Workshop on Wireless Sensor Networks*, (Piscataway, NJ, USA), pp. 354–365, IEEE Press, 2005.

[230] P. J. Marrón, M. Gauger, A. Lachenmann, D. Minder, O. Saukh, and K. Rothermel, "FlexCup: A Flexible and Efficient Code Update Mechanism for Sensor Networks," in *Proceedings of the 3rd European Conference on Wireless Sensor Networks*, EWSN'06, (Berlin, Heidelberg), pp. 212–227, Springer-Verlag, 2006.

[231] A. Mtibaa and S. Mastorakis, "NDNTP: A Named Data Networking Time Protocol," *IEEE Network*, vol. 34, pp. 235–241, September 2020.

[232] P. Kietzmann, L. Boeckmann, L. Lanzieri, T. C. Schmidt, and M. Wählisch, "A Performance Study of Crypto-Hardware in the Low-end IoT," in *International Conference on Embedded Wireless Systems and Networks (EWSN'21)*, (New York, USA), ACM, February 2021.

[233] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," RFC 8032, IETF, January 2017.

[234] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, IETF, February 1997.

[235] C. Bellman and P. C. van Oorschot, "Analysis, Implications, and Challenges of an Evolving Consumer IoT Security Landscape," in *17th International Conference on Privacy, Security and Trust (PST)*, pp. 1–7, IEEE, August 2019.

[236] C. Gündogan, P. Kietzmann, M. S. Lenders, H. Petersen, M. Frey, T. C. Schmidt, F. Shzu-Juraschek, and M. Wählisch, "The Impact of Networking Protocols on Massive M2M Communication in the Industrial IoT," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 18, pp. 4814–4828, Dec. 2021.

[237] T. Schmidt, M. Waehlisch, and G. Fairhurst, "Multicast Mobility in Mobile IP Version 6 (MIPv6): Problem Statement and Brief Survey," RFC 5757, IETF, February 2010.

[238] H. Kwon, J. Park, and N. Kang, "Challenges in Deploying CoAP Over DTLS in Resource Constrained Environments," in *Information Security Applications*, (Cham, Switzerland), pp. 269–280, Springer, 2016.

[239] E. Rescorla, H. Tschofenig, T. Fossati, and A. Kraus, "Connection Identifiers for DTLS 1.2," Internet-Draft – work in progress 13, IETF, June 2021.

[240] J. Schaad, "CBOR Object Signing and Encryption (COSE)," RFC 8152, IETF, July 2017.

[241] C. Amsuess, J. Mattsson, and G. Selander, “CoAP: Echo, Request-Tag, and Token Processing,” Internet-Draft – work in progress 14, IETF, October 2021.

[242] M. Vucinic, G. Selander, J. Mattsson, and D. Garcia-Carillo, “Requirements for a Lightweight AKE for OSCORE,” Internet-Draft – work in progress 04, IETF, June 2020.

[243] A. Rahman and E. Dijk, “Group Communication for the Constrained Application Protocol (CoAP),” RFC 7390, IETF, October 2014.

[244] M. Tiloca, G. Selander, F. Palombini, J. Mattsson, and J. Park, “Group OSCORE - Secure Group Communication for CoAP,” Internet-Draft – work in progress 14, IETF, March 2022.

[245] D. Whiting, R. Housley, and N. Ferguson, “Counter with CBC-MAC (CCM),” RFC 3610, IETF, September 2003.

[246] J. Mattsson, J. Fornehed, G. Selander, F. Palombini, and C. Amsuess, “Controlling Actuators with CoAP,” Internet-Draft – work in progress 06, IETF, September 2018.

[247] D. McGrew and D. Bailey, “AES-CCM Cipher Suites for Transport Layer Security (TLS),” RFC 6655, IETF, July 2012.

[248] J. Mattsson and D. Migault, “ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2,” RFC 8442, IETF, September 2018.

[249] G. Selander, J. Mattsson, and F. Palombini, “Ephemeral Diffie-Hellman Over COSE (EDHOC),” Internet-Draft – work in progress 01, IETF, March 2020.

[250] D. McGrew, “An Interface and Algorithms for Authenticated Encryption,” RFC 5116, IETF, January 2008.

[251] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig, “Transport Layer Security (TLS) Session Resumption without Server-Side State,” RFC 5077, IETF, January 2008.

[252] Y. Yu, A. Afanasyev, D. Clark, kc claffy, V. Jacobson, and L. Zhang, “Schematizing Trust in Named Data Networking,” in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, ICN ’15, (New York, NY, USA), pp. 177–186, ACM, 2015.

[253] A. Compagno, M. Conti, and R. Droms, “OnboardICNg: a Secure Protocol for Onboarding IoT Devices in ICN,” in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ICN ’16, (New York, NY, USA), pp. 166–175, ACM, 2016.

[254] T. Mick, R. Tourani, and S. Misra, “LASeR: Lightweight Authentication and Secured Routing for NDN IoT in Smart Cities,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 755–764, 2018.

[255] M. Bellare and P. Rogaway, “Entity Authentication and Key Distribution,” in *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO ’93, (Berlin, Heidelberg), pp. 232–249, Springer, 1993.

[256] F. Bersani and H. Tschofenig, “The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method,” RFC 4764, IETF, January 2007.

[257] T. Narten, R. Draves, and S. Krishnan, “Privacy Extensions for Stateless Address Autoconfiguration in IPv6,” RFC 4941, IETF, September 2007.

[258] R. Hinden and S. Deering, “IP Version 6 Addressing Architecture,” RFC 2373, IETF, July 1998.

[259] M. Tiloca and E. Dijk, “Proxy Operations for CoAP Group Communication,” Internet-Draft – work in progress 06, IETF, March 2022.

[260] M. Tiloca, R. Hoeglund, C. Amsuess, and F. Palombini, “Observe Notifications as CoAP Multicast Responses,” Internet-Draft – work in progress 04, IETF, November 2020.

[261] C. Amsuess and M. Tiloca, “Cacheable OSCORE,” Internet-Draft – work in progress 05, IETF, July 2022.

[262] J. Mattsson and F. Palombini, “Comparison of CoAP Security Protocols,” Internet-Draft – work in progress 01, IETF, March 2018.

[263] F. Palombini, M. Tiloca, R. Hoeglund, S. Hristozov, and G. Selander, “Combining EDHOC and OSCORE,” Internet-Draft – work in progress 01, IETF, November 2020.

[264] C. Gündogan, P. Kietzmann, T. C. Schmidt, and M. Wählisch, “ICNLoWPAN – Named-Data Networking in Low Power IoT Networks,” in *Proc. of 18th IFIP Networking Conference*, (Piscataway, NJ, USA), pp. 1–9, IEEE Press, May 2019.

[265] I. Psaras, W. K. Chai, and G. Pavlou, “Probabilistic In-network Caching for Information-centric Networks,” in *Proc. of the second ICN workshop on Information-centric networking*, (New York, NY, USA), pp. 55–60, ACM, 2012.

[266] N. Fotiou, H. Islam, D. Lagutin, T. Hakala, and G. C. Polyzos, “CoAP over ICN,” in *Proc. of IFIP NTMS*, (Piscataway, NJ, USA), pp. 1–4, IEEE, 2016.

[267] K. Pentikousis, B. Ohlman, E. Davies, S. Spirou, and G. Boggia, “Information-Centric Networking: Evaluation and Security Considerations,” RFC 7945, IETF, September 2016.

[268] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin, “An Information Centric Network for Computing the Distribution of Computations,” in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ICN ’14, (New York, NY, USA), pp. 137–146, ACM, 2014.

[269] M. Król, K. Habak, D. Oran, D. Kutscher, and I. Psaras, "RICE: Remote Method Invocation in ICN," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, ICN '18, (New York, NY, USA), pp. 1–11, ACM, 2018.

[270] P. Gauthier, J. Cohen, M. Dunsmuir, and C. Perkins, "Web Proxy Auto-Discovery Protocol," Internet-Draft – work in progress 01, IETF, July 1999.

[271] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest Flooding Attack and Countermeasures in Named Data Networking," in *Proc. of IFIP Networking*, (Piscataway, NJ, USA), IEEE Press, 2013.

[272] S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 85–110, 1990.

[273] E. Dijk, C. Wang, and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)," Internet-Draft – work in progress 07, IETF, July 2022.

[274] T. Schmidt, M. Waehlisch, and S. Krishnan, "Base Deployment for Multicast Listener Support in Proxy Mobile IPv6 (PMIPv6) Domains," RFC 6224, IETF, April 2011.

[275] T. Schmidt, S. Gao, H. Zhang, and M. Waehlisch, "Mobile Multicast Sender Support in Proxy Mobile IPv6 (PMIPv6) Domains," RFC 7287, IETF, June 2014.

[276] H. he and B. Chen, "An Elliptic Curve Based Name Privacy Protection Mechanism for Sensory Data Centric Named Data Networking," in *Proc. of 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, (Piscataway, NJ, USA), pp. 56–62, IEEE, 2019.

[277] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda, "Privacy Risks in Named Data Networking: What is the Cost of Performance?," *SIGCOMM Comput. Commun. Rev.*, vol. 42, pp. 54–57, September 2012.

[278] I. Moiseenko, L. Wang, and L. Zhang, "Consumer / Producer Communication with Application Level Framing in Named Data Networking," in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, ICN '15, (New York, NY, USA), pp. 99–108, ACM, 2015.

[279] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *Proc. of NSDI*, NSDI'04, (Berkeley, CA, USA), p. 2, USENIX Association, 2004.

[280] S. Deering, "Host extensions for IP multicasting," RFC 1112, IETF, August 1989.

[281] P. V. Mieghem, G. Hooghiemstra, and R. van der Hofstad, "On the Efficiency of Multicast," *IEEE Transactions on Networking*, vol. 9, no. 6, pp. 719–732, 2001.

[282] LoRa Alliance – Technical Committee, “Lorawan 1.1 specification,” tech. rep., LoRa Alliance, Oct. 2017.

[283] Bluetooth Special Interest Group, “Bluetooth Core Specification,” Bluetooth Specification 5.1, Bluetooth SIG, January 2019.

[284] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby, and C. Gomez, “IPv6 over BLUETOOTH(R) Low Energy,” RFC 7668, IETF, October 2015.

[285] I. Wijnands, E. Rosen, A. Dolganow, T. Przygienda, and S. Aldrin, “Multicast Using Bit Index Explicit Replication (BIER),” RFC 8279, IETF, November 2017.

[286] J. Hui and R. Kelsey, “Multicast Protocol for Low-Power and Lossy Networks (MPL),” RFC 7731, IETF, February 2016.

[287] O. Bergmann, C. Bormann, S. Gerdes, and H. Chen, “Constrained-Cast: Source-Routed Multicast for RPL,” Internet-Draft – work in progress 01, IETF, October 2017.

[288] P. Thubert, “RPL-BIER,” Internet-Draft – work in progress 02, IETF, July 2018.

[289] E. Dijk, C. Wang, and M. Tiloca, “Group Communication for the Constrained Application Protocol (CoAP),” Internet-Draft – work in progress 02, IETF, November 2020.

[290] M. Tiloca and E. Dijk, “Proxy Operations for CoAP Group Communication,” Internet-Draft – work in progress 02, IETF, November 2020.

[291] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, “Named Data Networking of Things (Invited Paper),” in *Proc. of IEEE International Conf. on Internet-of-Things Design and Implementation (IoTDI)*, (Los Alamitos, CA, USA), pp. 117–128, IEEE Computer Society, 2016.

[292] C. Gündogan, C. Amsüss, T. C. Schmidt, and M. Wählisch, “Content Object Security in the Internet of Things: Challenges, Prospects, and Emerging Solutions,” *IEEE Transactions on Network and Service Management (TNSM)*, vol. 19, pp. 538–553, March 2022.

[293] T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” RFC 3986, IETF, January 2005.

[294] G. Selander, J. Mattsson, and F. Palombini, “Ephemeral Diffie-Hellman Over COSE (EDHOC),” Internet-Draft – work in progress 15, IETF, July 2022.