FACULTY OF ENGINEERING
AND ARCHITECTURE

Multi-Level Interoperability in Heterogeneous Low Power Wide Area Networks

Bart Moons

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Information Engineering Technology

Supervisors
Prof. Jeroen Hoebeke, PhD - Prof. Eli De Poorter, PhD

Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

April 2022

GHENT
UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Multi-Level Interoperability in Heterogeneous Low Power Wide Area Networks

Bart Moons

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Information Engineering Technology

Supervisors
Prof. Jeroen Hoebeke, PhD - Prof. Eli De Poorter, PhD

Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

April 2022

GHENT UNIVERSITY

ISBN 978-94-6355-585-2
NUR 986
Wettelijk depot: D/2022/10.500/26

Members of the Examination Board

Chair

Prof. Patrick De Baets, PhD, Ghent University

Other members entitled to vote

Prof. Pieter Colpaert, PhD, Ghent University
Prof. Ingrid Moerman, PhD, Ghent University
Prof. Laurent Toutain, PhD, Institut Mines-Télécom Atlantique, France
Prof. Bruno Volckaert, PhD, Ghent University
Prof. Maarten Weyn, PhD, Universiteit Antwerpen

Supervisors

Prof. Jeroen Hoebeke, PhD, Ghent University
Prof. Eli De Poorter, PhD, Ghent University

"Patience is also a form of action."

Auguste Rodin

When I remencise about the past four years, I realize how many people were involved in shaping me and this dissertation into its current form. Therefore I would like to dedicate a few words to those who supported me throughout the years.

First and foremost, I would like to thank my supervisors Jeroen Hoebeke and Eli De Poorter. They believed in me and gave me the opportunity to start on an adventure I didn't think I could complete. I had the opportunity to work closely with Jeroen and want to thank him in particular for his support. He advised me when I was stuck on a problem and enlightened me during discussions. He showed me new ways of tackling problems, presenting the solutions in a lightly digestible way and writing them down clearly and concisely. He also gave me the opportunity to steer my way and work on topics that were less related to the projects. Thank you Jeroen and Eli.

At this point I would also like to thank the examination board for taking the time to read this dissertation, providing constructive feedback and being part of the examination board. Thank you Laurent Toutain, Maarten Weyn, Bruno Volckaert, Pieter Colpaert, Ingrid Moerman and Patrick De Baets.

A large part of the PhD work was done from my home office, however before the unfortunate turn of events in early 2020 I was lucky enough to share my office space on the 11th floor in the iGent tower with many great colleagues. I can think of numerous lunches, coffee breaks and hallway conversations that inspired me and/or made my day. Thank you Mathias, Robbe, Jaron, Michael, Matteo, Jen, Adnan, Amina, Andy, Subho, Jetmir, Abdulkadir, Vasilis, Nicola, Dries, Felipe, Merkebu, Ben, Jan, Pieter, Pablo, Spilios, Irfan, Bart, Jono, Vincent, Muhammad, Xianjun, Wei. In particular, I would like to thank my close colleagues Jetmir Haxhibeqiri and Abdulkadir Karaagac. During our discussion they showed me how to have a critical mind and find practical solutions to technical problems. They gave me advise on the topics that were related to mine and provided priceless input. Thank you guys.

Even before starting my PhD, people were important that lead me to this opportunity. During my master's thesis I had the opportunity to work with Peter Hellinckx on the reprogramming of embedded systems. If he hadn't introduced me to Eli and Jeroen, I wouldn't be sitting here writing this book. Thank you Peter.

Not only did my colleagues contributed in getting this book of the press, so did my f(r)amily. During my PhD studies I was surrounded at home by three wonderful guys. They made sure, at appropriate and inappropriate

moments, to distract my mind and showed that in addition to effort, there is also pleasure. Thank you Cedric, Anthony and Willi.

Finally I would like to thank my parents, aunts and brothers for the support they gave me throughout the years. Thank you papa for the opportunities you have given me, they have shaped me in many ways. Thank you Christel for the conversations and discussions, your look at the world gives me a broader perspective. Thank you Tante Neeltje for supporting my creativity and my interests in everything other than science. Thank you mama for your love and endless support, you pulled me through at times I thought about giving up. Thank you Dirk for your clarity, you can make it easy for me to understand topics I know little about. Thank you Daan, Cota, Tom and Yanne for being my brothers and sisters. Thank you Ella for showing the sense of wonder at life that we sometimes tend to forget.

Ghent, March 2022
Bart Moons

Table of Contents

List of Figures

List of Tables

List of Acronyms

0-9

3GPP 3rd Generation Partnership Project
6LoWPAN IPv6 over LoWPANs

A

ADR Adaptive Data Rate
AI Artificial Intelligence
AMQP Advanced Message Queuing Protocol
AMS Administration Management Server
AoA Angle of Arrival
AP Anchor Point
API Application Programming Interface
ARO Address Registration Option
ARP Address Resolution Protocol
ARPANET Advanced Research Projects Agency Network
AVS Alexa Voice Service
AWS Amazon Web Service

B

BDS BeiDou Navigation Satellite System
BLE Bluetooth Low Energy
BPSK Binary Phase Shift Keying

BS Base Station

C

CAO Context Advertisement Object
CBOR Concise Binary Object Representation
CDA Compression Decompression Action
CGI Cell Global Identity
CoAP Constrained Application Protocol
CORECONF CoAP Management Interface
COST European Cooperation in Science and Technology
COTS Commercial Off The Shelf
CCN Content Centric Networking
CPU Central Processing Unit
CR Coding Rate
CRC Cyclic Redundancy Check
CRI Container Runtime Interface
CRUD Create, Read, Update, Delete
CSMA/CA Carrier-Sense Multiple Access with Collision Avoidance
CSS Chirp Spread Spectrum
CYBER Cyber Security for Consumer Internet of Things

D

D7A DASH-7 Alliance
DAD Duplicate Address Detection
DARPA Department of Defense's Advanced Research Project's Agency
DNS Domain Name System
DRX Discontinuous Reception
DTag Datagram Tag
DTLS Datagram Transport Layer Security

E

EC-GSM Extended Coverage Global System for Mobile Communication (GSM)

eDRX extended Discontinuous Reception (DRX)

EH Extended Header

eNB Evolved Node B

eMTC enhanced Machine Type Communication

ETSI European Telecommunications Standards Institute

EUI Extended Unique Identifiers

F

FCC Federal Communications Commission

FCN Fragment Count Number

FEC Forward Error Correction

FL Field Length

FP Field Position

FSM Finite State Machine

G

GB Gigabyte

GFSK Gaussian Frequency Shift Keying

GHz Gigahertz

GLONASS Global Navigation Satellite System

GNSS Global Navigation Satellite System

GPS Global Positioning System

GSM Global System for Mobile Communication

GUI Graphical User Interface

GW Gateway

H

HTML Hyper Text Markup Language
HTTP Hyper Text Transfer Protocol

I

ICMP Internet Control Message Protocol
ICMPv6 ICMP for IPv6
ICN Information Centric Network
IEC International Electrotechnical Commission
IEEE Institute of Electrical and Electronics Engineers
IETF Internet Engineering Task Force
IGS International GNSS Service
IoT Internet of Things
I/O Input/Output
IP Internet Protocol
IPv4 Internet Protocol Version 4
IPv6 Internet Protocol Version 6
IPHC Internet Protocol Header Compression
IRNSS Indian Regional Navigation Satellite System
ISM Industrial, Scientific and Medical
ISO International Organization for Standardization
IT Information Technology

J

JSON JavaScript Object Notation
JSON-LD JavaScript Object Notation (JSON)-Linked Data (LD)

K

kB kilobyte

kbps kilobit per second

***k*NN** k Nearest Neighbors

L

LAN Local Area Network

LBS Location-based Service

LD Linked Data

LDP Linked Data Platform

LOD Linked Open Data

LoS Line-of-Sight

LoRa Long Range

LoRaWAN Long Range Wide Area Network

LoWPAN Low-Power Wireless Personal Area Network

LPG Labeled Property Graph

LPWAN Low Power Wide Area Network

LTE Long Term Evolution

LTE-M Long Term Evolution (LTE)-Machine Type Communication (MTC)

LwM2M Light Weight Machine to Machine

M

MAC Medium Access Control

MCU Microcontroller Unit

MERIT Michigan Educational Research Information Triad

MHz Mega Hertz

MMU Memory Management Unit

MO Matching Operator

MQTT Message Queuing Telemetry Transport

MTC Machine Type Communication

MTU Maximum Transfer Unit

N

NA Neighbor Advertisement
NAT Network Address Translation
NB-IoT Narrowband IoT
NCE Neighbor Cache Entry
NCP Network Control Program
ND Neighbor Discovery
NDN Named Data Networking
NETCONF Network Configuration Protocol
NGSI Next Generation Service Interfaces
NGSI-LD Next Generation Service Interfaces-Linked Data
NHC Next Header Compression
NIC Network Interface Card
NLoS Non-Line-of-Sight
NS Network Server
ntpd Network Time Protocol daemon
NUD Neighbor Unreachability Detection

O

OMA Open Mobile Alliance
OOK On-Off Keying
OS Operating System
OSI Open Systems Interconnection
OTA Over The Air
OTAU Over The Air (OTA) Update
OTDoA Observed Time Difference of Arrival
OWL Web Ontology Language

P

PDU Protocol Data Unit

PER Packet Error Ratio
PG Property Graph
PIO Prefix Information Option
PNT Positioning, Navigation and Timing
POD Personal Online Datastore
PPP Precise Point Positioning
PSM Power Saving Mode

Q

QoS Quality of Service

R

RA Router Advertisement
RAM Random Access Memory
RAT Radio-Access Technology
RCS Reassembly Check Sequence
RDF Resource Description Framework
REST Representational State Transfer
RFC Request For Comments
RFID Radio Frequency Identification
RML RDF Mapping Language
ROM Read Only Memory
RPC Remote Procedure Call
RPL IPv6 Routing Protocol for Low-power and Lossy Networks
RS Router Solicitation
RSS Received Signal Strength
RTS/CTS Request to Send and Clear to Send

S

SAREF Smart Applications REFerence

SAVI Source Address Validation Improvement
SCHC Static Context Header Compression
SCM SCHC Context Manager
SF Spreading Factor
SID Schema Item Identifiers
SLLAO Source Link-Layer Address Option
SNMP Simple Network Management Protocol
SNR Signal-to-Noise Ratio
SoC System on Chip
SOTA State of the Art
SPARQL SPARQL Protocol and RDF Query Language
SPO SCHC Parameter Option
SRD Short Range Device
SRR SCHC Rule Registry
SSN Semantic Sensor Networks
SSH Secure Shell
SOSA Sensor, Observation, Sample and Actuator

T

TCP Transmission Control Protocol
TDoA Time Difference Of Arrival
TD Thing Description
TLS Transport Layer Security
TLV Type-Length Value
Turtle Terse RDF Triple Language
TV Target Value

U

UCL University College London
UDP User Datagram Protocol
UE User Equipment

UNB Ultra Narrowband
UWB Ultra Wideband
URI Uniform Resource Identifier
URN Uniform Resource Name

V

VDM Virtual Device Manager
VNO Virtual Network Operator

W

W3C World Wide Web Consortium
WG Working Group
WoT Web of Things
WPAN Wireless Personal Area Network
WS Web Sockets
WSN Wireless Sensor Network
WWW World Wide Web

X

XML eXtensible Markup Language

Y

YANG Yet Another Next Generation

Samenvatting

Sinds de mens begon met het uitvinden van motoren en machines, deed hij dit steeds met het oog op het verbeteren van de kwaliteit van zijn leven, zijn manier van werken en zijn welzijn in het algemeen. Aangezien veel taken die de mens bezig houdt getypeerd worden door herhaling, is de mogelijkheid van de computer om repetitieve taken over te nemen de drijvende kracht achter hun ontwikkeling geweest. Het was al in 1820 toen Charles Babbage de Differentiaal Machine bedacht om tabellen van veeltermen uit te rekenen. 80 jaar later was het Konrad Zuse die de eerste functionele mechanische computer uitvond die zwevendekommagetallen kon optellen, aftrekken, vermenigvuldigen, delen en daarvan een vierkantswortel kon nemen. Sindsdien zijn computers steeds kleiner geworden, is hun rekenkracht blijven toenemen en zijn ze begonnen met elkaar te communiceren.

Veel van de vooruitgang in de informatietechnologie is iets wat we te danken hebben aan de mogelijkheid van computers om met elkaar te communiceren. Sterker nog, het is niet de communicatieverbinding van een computer die hem krachtig maakt, het is zijn vermogen om andere systemen te begrijpen in een ecosysteem van apparaten, diensten en technologieën. Het door anderen volledig begrepen worden, wordt aangeduid met interoperabiliteit en is één van de concepten die de hoekstenen van het Internet en het Wereld Wijde Web (WWW) hebben gelegd.

Eén van de zaken waar de afgelopen jaren veel aandacht aan interoperabiliteit is besteed, is het Internet der Dingen (IdD). Het IdD wordt gekenmerkt door zeer kleine toestellen die overal geïnstalleerd kunnen worden om informatie over hun omgeving te verstrekken. Hun vermogen om informatie over de ether te sturen, met behulp van technologieën als Bluetooth en IEEE 802.15.4, laat hen toe om de mens bij te staan in zijn dagdagelijkse taken zoals het regelen en opvolgen van mobiliteit, domotica en gezondheidszorg.

Het IdD vond niet alleen zijn toepassing in korte-afstandsscenario's. Verschillende andere domeinen, zoals landbouw, milieumonitoring, slimme steden en logistiek vereisten draadloze communicatie over lange afstanden met een laag stroomverbruik en leidden tot de ontwikkeling van draadloze langeafstands-technologieën zoals Long Range (LoRa), Sigfox, DASH-7 en Narrowband IoT (NB-IoT). Veel van deze technologieën maken gebruik van de niet-gelicentieerde sub-Gigahertz (GHz) frequentiebanden. Dit deel van het radiofrequentiespectrum helpt om hun bereik te vergroten, maar beperkt de bandbreedte voor gegevensdoorvoer. Hierdoor wordt de integratie met

bestaande netwerken bemoeilijkt, aangezien bestaande protocollen werden ontwikkeld voor machines die niet de beperkingen hadden die kenmerkend zijn voor draadloze langeafstands-technologieën. Daarom zal dit doctoraat interoperabiliteit van draadloze langeafstands-netwerken met het Internet en het Web bekijken. Het zal dit doen door te focussen op 5 niveaus van interoperabiliteit.

De netwerken die het IdD verbinden, zijn en zullen grotendeels heterogeen blijven. Zelfs in het subdomein van draadloze langeafstands-netwerken bestaat er een enorme variëteit. Het verbinden van heterogene netwerken is iets waar het Internet van het begin af aan mee te kampen had. Om dit op te lossen wordt gebruik gemaakt van een meerlagige aanpak, beter bekend als het Open Systems Interconnection (OSI)-model. De smalle taille van het Internet model gebruikt het Internet Protocol (IP) om apparaten te adresseren en pakketten in een netwerk af te leveren. Daarbovenop kan het verbindingsgeoriënteerde Transmission Control Protocol (TCP) gebruikt worden om betrouwbaarheid te bieden of het User Datagram Protocol (UDP) voor onbetrouwbare, snelle(re) transmissies. Deze protocollen zijn echter ontworpen voor netwerken met hoge bandbreedte en zijn niet geschikt voor de technologieën met lage bandbreedte die kenmerkend zijn voor draadloze langeafstands-netwerken.

Het eerste niveau van interoperabiliteit - *netwerk-interoperabiliteit* - probeert dergelijke heterogene netwerken op een uniforme manier met elkaar te verbinden. Om dit te doen, richt dit doctoraat zich op apparaten die meerdere draadloze communicatietechnologieën ondersteunen. De belangrijkste uitdaging die op dit niveau wordt aangepakt, is de herbruikbaarheid van applicaties over deze verschillende draadloze technologieën. Met behulp van het nieuwe Static Context Header Compression (SCHC) protocol van het Internet Engineering Task Force (IETF), kunnen deze apparaten en netwerken toch de bekende en wijdverbreide internettalen ondersteunen. Dit wordt bereikt door compressie van de header-informatie van de protocollen van de bovenste lagen en door fragmentatie van grote internetpakketten die niet kunnen worden getransporteerd over technologieën die een maximale pakketgrootte kunnen hebben tot amper 8 bytes.

De SCHC standaard gaat ervan uit dat een statische context werd gedistribueerd voor het uitrollen van het netwerk. Deze context wordt aan beide zijden van het netwerk gebruikt om de headers van de bovenste lagen te comprimeren en te decomprimeren. De manier om dit te doen - aan de hand van een middleware die zich tussen twee netwerken bevindt - is niet helemaal in lijn met het idee van het Internet dat oppert om de intelligentie van het netwerk aan de rand te houden. Daarom stelt dit doctoraat twee apparaatregistratie- en context-configuratiemechanismen voor SCHC-netwerken voor. Deze oplossingen kunnen worden gebruikt om middleware te vervangen zodat eenvoudigere upgrades van het netwerk en de evolutie van applicaties mogelijk worden. Bijgevolg kunnen complexe Application Programming Interfaces (APIs) worden uitgesloten om tot echte *netwerk- interoperabiliteit*

in draadloze langeafstands-netwerken te komen. Beide mechanismen bieden ook een dynamische oplossing die gebruikt kan worden om protocollen van de hogere lagen te configureren om de zendtijd te beperken en zo energie te besparen.

Een tweede niveau van interoperabiliteit dat werd onderzocht in dit proefschrift, werd gedefinieerd als *apparaat interoperabiliteit*. Dit heeft tot doel zowel hoogwaardige apparaten als laagwaardige apparaten met elkaar te verbinden. Aangezien het IdD is samengesteld uit talloze apparaten die verschillende mogelijkheden hebben op het vlak van verwerkingskracht, opslagruimte, batterijcapaciteit en communicatietechnologie, wordt een oplossing voorgesteld om hoogwaardige systemen en kleine sensorapparaten, met slechts 20 kilobytes (kBs) Random Access Memory (RAM) en 192 kBs Read Only Memory (ROM), met elkaar te verbinden. Om dit op te lossen stellen we een modulaire, schaalbare en flexibele architectuur voor. Met de voorgestelde architectuur is het mogelijk om verkeer van hoogwaardige netwerken naar heterogene, laagwaardige draadloze netwerken te routeren. Bovendien zorgt de architectuur ervoor dat multimodale apparaten kunnen overschakelen tussen verschillende netwerken zonder hun status te verliezen. Aan de kant van het apparaat is daarvoor een netwerk-schakelalgoritme ontwikkeld, waardoor sensoren efficiënt meerdere langeafstands-technologieën kunnen gebruiken om met de beperkingen van elkaar om te gaan.

Het derde interoperabiliteitsniveau dat behandeld wordt in dit onderzoek, is *semantische interoperabiliteit*, een trend die zich voornamelijk op het WWW voordoet. Semantische interoperabiliteit wordt door het World Wide Web Consortium (W3C) gedefinieerd als "verschillende services en applicaties in staat stellen om informatie, gegevens en kennis op een zinvolle manier uit te wisselen, zowel op als buiten het web". De belangrijkste drijfveer achter de fragmentatie op het huidige web, zoals aangegeven door het W3C, is het blootleggen van gegevens met behulp van propriëtaire APIs. De gegevens die worden gegenereerd en weergegeven door APIs zijn meestal geformatteerd met behulp van bekende gegevensindelingen, zoals JavaScript Object Notation (JSON), Concise Binary Object Representation (CBOR) of eXtensible Markup Language (XML), maar de gegevensmodellen en -schema's zijn meestal eigendomsmatig. Daarom is het voor machines vaak niet mogelijk om gegevens uit deze APIs te lezen en te verwerken omdat er geen betekenis aan is verbonden. Daarom bestudeerde dit onderzoek de haalbaarheid van het gebruik van semantische webtechnologieën in heterogene langeafstands-technologieën. Zoals onze studie zal aantonen, is het onpraktisch om Resource Description Framework (RDF)-gegevens te gebruiken over dergelijke netwerken. Daarom wordt een lichter alternatief gebruikt dat zowel voor apparaat-beheer als voor het ophalen van gegevens kan worden gebruikt. Om uiteindelijk tot semantische interoperabiliteit te komen, wordt een architectuur voorgesteld die gegevens tussen verschillende ontologieën in kaart brengt.

Ten slotte is het voor applicatieontwikkelaars vaak ingewikkeld om *plat-*

formoverschrijdende en *domeinoverschrijdende interoperabiliteit* te realiseren in het huidige IdD-ecosysteem vanwege de overvloed aan beschikbare besturingssystemen, programmeertalen, architecturen en toegangsmechanismen. Om de integratie-inspanningen te beperken en tot dit niveau van interoperabiliteit te komen, presenteert dit doctoraat een modulaire, flexibele en schaalbare netwerkarchitectuur. Dankzij het modulaire ontwerp kunnen applicatieontwikkelaars stromen programmeren tussen componenten van bestaande applicaties in eender welke programmeertaal. Door middel van een berichtenkanaal kunnen componenten met elkaar worden verbonden, wat de flexibiliteit en schaalbaarheid vergroot. Verder wordt een algemeen berichten-patroon, volgens een zeer eenvoudig schema, in JSON gebruikt om communicatie tussen componenten en *syntactische interoperabiliteit* te bieden.

Samengevat stelt dit proefschrift een aantal oplossingen voor die interoperabiliteit biedt van sensor netwerk tot data opslag voor heterogene langeafstands-netwerken. Het onderzoek toont aan dat de huidige SCHC-standaard netwerkinteroperabiliteit biedt tussen langeafstands-compatibele componenten. Om netwerkinteroperabiliteit tussen de eigenlijke netwerken en het Internet te bieden, blijkt dat de huidige specificatie moet worden uitgebreid. Ten tweede wil dit doctoraat door middel van een middleware-architectuur zowel laagwaardige als hoogwaardige apparaten verbinden, waarbij het een manier biedt om te schakelen tussen heterogene langeafstands-netwerken. Ten slotte wordt in een latere fase de middleware uitgebreid om systemen van meerdere domeinen te koppelen. Hoewel volledig interoperabele langeafstands-systemen nog niet voor vandaag zijn, biedt dit proefschrift een kleine stap in de richting van een Internet-compatibel Web of Things.

Summary

Ever since humans started crafting engines and machines, they have been doing so to improve their way of living, working and being in general. Since many tasks that keep humans occupied are characterized by repetition, the driving force behind the development of computers has been their ability to automate repetitive tasks. It was way back in 1820 when Charles Babbage conceptualized the Difference Engine to calculate polynomials and 80 years later, Konrad Zuse invented the first functional mechanical computer that could add, subtract, multiply, divide and take a square root of floating point numbers. Since then, computers have been decreasing in size, increasing in processing power and started communicating with each other.

Many of the advancements in Information Technology (IT) are something we owe to the ability of computers to communicate with each other. Even more, it is not only a machine's communication link that makes it powerful, it is its ability to understand others in an ecosystem of devices, services and technologies. The concept of interfaces that are completely understood by others is referred to as interoperability and is one of the concepts that has laid the cornerstones of the Internet and the World Wide Web (WWW).

One of the things that increasingly gained attention around interoperability in recent years - due to its infancy - is the Internet of Things (IoT). The IoT is characterized by very small devices that can be installed anywhere to provide information about their environment. Their ability to distribute information over the air, using technologies such as Bluetooth and IEEE 802.15.4, allows them to assist humans in their day-to-day tasks in several domains such as mobility, home automation and healthcare among others.

However, not only did the IoT find its applications in short range scenarios, several cases, such as agriculture, environmental monitoring, smart cities and logistics required low power, wireless communication over long distances and gave rise to Low Power Wide Area Network (LPWAN) technologies such as Long Range (LoRa), Sigfox, DASH-7 and Narrowband IoT (NB-IoT). Many of these technologies make use of the unlicensed sub-Gigahertz (GHz) frequency bands. This part of the radio frequency spectrum helps to extend their range, but limits the bandwidth for data throughput. This complicates the integration with existing networks since existing protocols were initially developed for machines that did not have the limitations that characterize wireless long range technologies. Therefore this PhD will look at Internet and semantic Web compliance in LPWANs and will do so by focusing on 5

levels of interoperability.

The networks that connect the IoT are, and will continue to be, largely heterogeneous. Even in the sub-domain of LPWANs there is a great variety. Since its inception, connecting heterogeneous networks has been an obstacle for the Internet. In order to solve this, the Internet employs a multi-layered approach, better known as the Open Systems Interconnection (OSI)-model. The narrow waist of the Internet hourglass uses the Internet Protocol (IP) to address devices and deliver packets in a network. On top of this, the Transmission Control Protocol (TCP) can be used to provide reliability or the User Datagram Protocol (UDP) for unreliable, fast(er) transmissions. However, these protocols were designed for high-bandwidth networks and are not suitable for the low bandwidth technologies that characterize LPWANs.

The first level of interoperability, called *network interoperability*, tries to interconnect such heterogeneous networks. In order to do so, this PhD focuses on devices that support multiple wireless communication technologies. The main challenge that is targeted on this level is the re-usability of applications over different wireless technologies. Using a new Internet Engineering Task Force (IETF) protocol, Static Context Header Compression (SCHC), these devices and networks can support the well-known and widespread Internet languages. This is done by compressing the header information of the upper layer protocols and fragmentation of large Internet packets that can not be transported over technologies with a Maximum Transfer Unit (MTU) as small as 8 bytes.

SCHC assumes that a static context is distributed before deployment. This context is used on both sides of the network to compress and decompress the headers of the upper layers. The middleware that sits between both networks is not completely in line with the mature idea of the Internet Protocol Version 6 (IPv6) that suggests to keep the intelligence of the network at the edge. Therefore this PhD proposes two device registration and context configuration mechanisms for SCHC networks. These solutions can be used to exclude middleware and allow for simpler upgrades of the network and the evolution of applications. Consequently, complex Application Programming Interfaces (APIs) can be excluded in order to achieve true *network interoperability* in LPWANs. Both mechanisms also provide a dynamic solution for configurable upper layer protocols in order to save airtime, and hence energy.

The second level of interoperability is defined as *device interoperability* and aims to connect both high-end devices and low-end devices. Since the IoT is composed out of numerous devices that have different capabilities in terms of processing power, storage space, battery capacity and communication technology, a solution is proposed to interconnect high-end systems and small sensor devices with as little as 20 kilobytes (kBs) of Random Access Memory (RAM) and 192 kBs of Read Only Memory (ROM) available. In order to solve this, we propose a modular, scalable and flexible architecture. With the proposed architecture it is possible to route traffic

from high-end networks towards heterogeneous constrained wireless networks. Moreover, the architecture allows multimodal devices to switch between different networks while maintaining their state. On the device side, a network switching algorithm was developed, enabling sensors to use multiple LPWAN technologies in order to cope with the limitations of one another.

A third level of interoperability that this research covers is *semantic interoperability*, a trend that is moving the WWW forward. It is defined by the World Wide Web Consortium (W3C) as "enabling different agents, services, and applications to exchange information, data and knowledge in a meaningful way, on and off the Web". The main driver behind the fragmentation on the current Web, as stated by the W3C, is the exposing of things and data using proprietary APIs. The data generated and exposed by APIs are typically formatted using well-known data formats, such as JavaScript Object Notation (JSON), Concise Binary Object Representation (CBOR) or eXtensible Markup Language (XML), but the data models and schemes are usually proprietary. Hence, it is often not possible for machines to read and process data from these APIs as no meaning is associated with it. Therefore, this research studied the feasibility of using semantic Web technologies in heterogeneous LPWANs. As our study will show, it is impractical to use Resource Description Framework (RDF) data over such constrained links. Therefore, a light weight alternative is employed that can be used for both management and data retrieval. In order to to create correspondences that will enable semantic interoperability, an architecture is proposed that maps data between different vocabularies.

Finally, achieving *cross-platform* and *cross-domain interoperability* in the current IoT ecosystem is complicated for application developers due to the plethora of available Operating Systems (OSs), programming languages, architectures and access mechanisms. In order to limit integration effort and provide this level of interoperability, this PhD presents a modular, flexible and scalable network architecture. The modular design allows application developers to build flows between components from existing applications in any programming language. Components can be interconnected by means of a message bus, which increases the flexibility and ability to scale. Furthermore, a common messaging pattern in JSON that follows a very basic scheme is used for communication between components in order to provide *syntactical interoperability*.

To conclude, this dissertation offers a solution that provides interoperability from sensor network to data storage for heterogeneous LPWANs. The research shows that the current SCHC standard provides network interoperability between LPWAN-enabled components. However, to provide network interoperability between LPWAN gateways and the Internet, it shows that the current specification can be extended. Secondly, by means of a middleware architecture, this PhD aims at connecting both low-end and high-end devices, while providing a way of switching between heterogeneous LPWANs. Finally, it presents an extended version of the middleware in order to connect

systems from multiple domains. Although completely interoperable LPWAN systems are still a long way from today, this dissertation provides a small step towards an Internet compliant Web of Things.

1 Introduction

"The Web as I envisaged it, we have not seen it yet. The future is still so much bigger than the past."

Tim Berners-Lee

1.1 Background

1.1.1 A Brief History of the Internet

Without the cables that connect the machines on the Internet, electronic devices wouldn't be considered smart. Since the ability to communicate with others is a prerequisite for these devices for being *smart*, they are tied - be it wired or wireless - to the cables that spread across our globe. It was as early as 1858 when vessels from both America and Britain laid a cable between their respective countries to place a cornerstone of the Internet. Even though the first Atlantic cable only remained in service a few days, the first instantaneous communication across the ocean was imminent. It was only in 1866 for a subsequent cable to endure the oceanic conditions and to become a successful communication tool that remained in use for nearly 100 years [1].

The first Atlantic cable had been in use as a telegraph wire system for ten years before Alexander Graham Bell patented the telephone, but it wasn't until 1956 for the first *wired* transatlantic phone call to occur. It had taken almost thirty years to develop technology that could boost voice signals to ensure that the volume of someone's voice did not diminish by the time it got the other end [2].

These advancements in wired communications laid the foundation of the Internet; a telecommunication system that allowed computers to communicate with one another over large distances. In 1969, AT&T and the Department of Defense's Advanced Research Project's Agency (DARPA) built a 50 kilobit per second (kbps) connection between mainframe computers at the University of Utah, the University of California - Los Angeles, the University of California - Santa Barbara and Stanford Research Institute, called Advanced Research Projects Agency Network (ARPANET). Hosts could establish links using the Network Control Program (NCP), a middle layer protocol stack that provided connections to other host computers and flow control between two nodes [3, 4]. The Request For Comments (RFC) concept was used to describe these protocols and has been in use ever since. The following years, other, foreign networks, such as the University College London (UCL), connected to the ARPANET and contributed to the first international heterogeneous computer network [5].

During those years, several other networks came into being, such as the Michigan Educational Research Information Triad (MERIT) network and the French CYCLADES, that explored alternatives for these packet switching networks. It soon became clear that a method was required to unify the different networking methods in order to interconnect them. Consequently, Kahn and Cerf published a protocol design that supported the sharing of resources in different packet switching networks [6]. This resulted in RFC 675 that solved many problems of interconnecting multiple computer networks [7].

By 1981, the ARPANET consisted of more than 200 hosts and became the technical core of what we know today as the Internet. By that time, the initial Transmission Control Protocol (TCP) protocol was split into the TCP and the Internet Protocol (IP) and replaced the obsolete NCP [8, 9].

These early standardization efforts were essential for the transition towards a global internet. The hardware-agnostic approach in TCP/IP and the interest in remote collaboration and exchange of data led to a spread of these technologies throughout the world.

1.1.2 The Web and the Internet of Things

Although collaboration and the sharing of information underlie the core values of the Internet, multiple protocols and programs existed at the application layer that were incompatible. The frustration of documentation systems that were unable to interchange information directly and the vision of an open information system led to the development of the World Wide Web (WWW) by Tim Berners Lee in 1989 [10]. The idea was similar to that of the internet: to provide a *decentralized*, interoperable communication system based on a set of protocols. The Hyper Text Transfer Protocol (HTTP), Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS) provided tools to create a collaborative space where everyone was able to participate in.

The rise of online services, such as YouTube, Facebook and eBay, shifted the original idea of the Web towards centralized, non-standardized information sharing. This is often referred to as the Web 2.0, describing websites that mainly host user-generated content. However, according to Cisco, there were already more connected devices than people somewhere between 2008 and 2009 [11]. This point, where the Web moved away from the *Internet of content* to the *Internet of data*, can be marked as the theoretical start of the Internet of Things (IoT).

The IoT is considered a collection of devices consisting of sensors and actuators that can exchange data and exert an influence on their environment due to their interconnectivity. The concept of connected, smart devices was discussed as early as 1982, when a modified Coca-Cola vending machine was connected to the ARPANET [1]. However, since most objects we want to connect do not have access to wired connectivity, wireless technologies are a key enabler for the IoT.

In 1985, the Federal Communications Commission (FCC) already admitted the need for such wireless connectivity by allowing spread spectrum systems to operate in parts of the Industrial, Scientific and Medical (ISM) bands (at 902-928, 2400-2483.5 and 5725-5850 Mega Hertz (MHz)) [12]. This laid the foundation for short range wireless technologies, such as Wi-Fi, Bluetooth, Radio Frequency Identification (RFID), ZigBee and IEEE 802.15.4. The latter two, both standardized in 2003, were developed with a focus on Wireless Personal Area Network (WPAN) devices that target low-cost, low-speed communication. A consequence of these technologies that use low bandwidth and low power communication to save energy, was that there was no physical support for the widespread internet protocols.

Even though the IP had been around for a few years, with major im-

[1] https://www.cs.cmu.edu/∼coke/

provements in version 6, getting it to such resource constrained devices was out of the question. Due to the overhead of the network and transport layers, the application layer was tied straight above the link layer. This contrasted with the end-to-end Internet vision, that strives to keep networks simple and generic by moving the intelligence (i.e. the protocol stack) to the edge. The problem that Kahn and Cerf had solved in 1974 for wired networks, now limited the integration of low throughput Wireless Sensor Networks (WSNs) into the Internet.

As a consequence, the Internet Engineering Task Force (IETF) started the development of the IPv6 over Low-Power Wireless Personal Area Networks (LoWPANs) (6LoWPAN) standard, which brought Internet Protocol Version 6 (IPv6) and User Datagram Protocol (UDP) connectivity to these sensor devices. This layer uses several optimizations, including header compression, to adapt the low bandwidth of WPANs to the higher bandwidth demands of the network layer. Yet, different application layer protocols made it again - just like in 1989 - difficult to interchange information directly. Therefore, a specialized protocol, the Constrained Application Protocol (CoAP), was built around the same principles as the HTTP and solved many of these interoperability issues which led to complete, integrable IoT solutions.

When these short range technologies climbed the *Slope of Enlightenment* to the *Plateau of Productivity* [2], many cases exposed themselves where these communication technologies did not suffice. Connecting monitoring tools to existing infrastructure over a vast environment, for example, requires wireless connectivity with transmission ranges of several kilometers. This led to the development of Low Power Wide Area Network (LPWAN) technologies such as Long Range (LoRa) and Sigfox that provide very long range communication, at the cost of a very low throughput (between 5 kbps and 0.1 kbps respectively, which is lower than the bandwidth of the initial ARPANET). These technologies have been optimized for battery-operated devices since the availability of power sources is either limited or the replacement of the battery is time consuming. Typical applications can therefore operate on a battery for five to ten years without human intervention. This, along with low hardware costs, significantly reduces the operation cost per unit and has contributed to the commercial success of LPWAN technologies [13]. An overview of popular LPWAN technologies and their applications is given in Table 1.1.

Consequently, large companies started rolling out LPWAN solutions. A subsidiary of the French company Veolia, for example, already operates 400.000 LoRa connected smart water meters and will roll out another 3 million. However, the worldwide installed number of 450 million LPWAN

[2] https://www.gartner.com/en/research/methodologies/gartner-hype-cycle

Table 1.1: LPWAN overview of data-rate, coverage and typical application areas. Technologies with ranges up to 2 kilometer are considered medium range technologies. Technologies beyond 2 kilometers, long range technologies.

Technology	Configuration	Coverage	Data-rate	Application
Sigfox	Uplink	5 - 25 km	0.1 kbps	Smart Grid, Logistics
	Downlink	5 - 25 km	0.6 kbps	
LoRa	SF 7	5 km	5.5 kbps	Agriculture, Smart Grid, Smart Cities
	SF 12	5 - 25 km	0.3 kbps	
DASH7	Lo-Rate	2 km	4.8 kbps	Industrial Automation, Automotive
	Normal	1 km	55.5 kbps	
	Hi-Rate	1 km	167 kbps	

compatible devices does not come anywhere near the 3 billion unlicensed LPWAN connections that were forecasted by the end of the year 2023 [14, 15].

One of the reasons for this rather low global adoption is - again - the lack of an open, standardized protocol stack. Currently, mainly proprietary application layer protocols are deployed that are adapted to the underlying technology, which limits the re-usability of applications. Apart from that, different deployments must be managed, secured and operated individually, which requires an engineered link towards every network. Even more, developing applications that are tied to the link layer cannot be easily ported towards other (wireless) technologies. Furthermore, due to the low bandwidth and duty cycle constraints of LPWANs, current solutions do not have support for Over The Air (OTA) Updates (OTAUs) and are therefore prone to runtime errors and security attacks. All these time consuming tasks make it difficult for (smaller) companies to develop sustainable LPWAN solutions. Therefore, novel mechanisms are required that support existing features from the Internet protocol stack like flow control and mechanisms to resume interrupted or corrupted transfers.

Out of these observations emerged the IETF LPWAN Working Group (WG). Their mission includes enabling IPv6 connectivity over LPWAN technologies and to optimize IPv6-based communications to the end devices. A solution was standardized in 2020 under the name of Static Context Header Compression (SCHC). The SCHC protocol is based on the assumption that in LPWANs mostly a static context will be used. Following this assumption, a shared context can be built between the LPWAN devices and the network side [16]. This context can be used to compress and decompress redundant

headers. In the best cases, only a single byte is required to represent the application layer headers, transport layer headers and network layer headers that can be used in the future to interconnect these technologies in a Web of Things [17].

1.2 The Future Web

The main objective in developing Internet of Things (IoT) applications is to integrate technology into our lives. Communication technologies such as LPWAN are powerful tools that provide context information about our environment. However, how this information is used in a human context, leaves much room for improvement. People are very often forced to change their context to fit the technological requirements, rather than technology adapting to people's context [18].

One of the main things that makes it difficult to integrate machines in a human context is the impossibility for machines to understand data. No common language is able to reveal the meaning of symbols to technology. However, just like human languages, the study of semantics can provide context information. Applying these concepts to the Web, will reveal the *Semantic Web*. The integration of all kinds of IoT devices on top of that will ultimately lead to our environment adapting to our needs.

Consequently, information about our living environment and our interactions within will be exposed towards actuators and Web applications with the intention to simplify our lives. However, as every Linux superuser knows, with great power comes great responsibility. Concerning privacy, it will be of great importance that users can control their data and the systems that are part of their lives. Therefore, the future web should work towards more fairness, where users are able to choose where to store their data, independent of the applications they use. As users maintain their data themselves, they are free to move to other "front-ends". Such a solution is currently being implemented by Solid, one of the largest personal data storage ecosystems that focuses on decentralization of social data using standardized vocabularies.

1.3 Research Challenges

This PhD dissertation addresses some of the challenges affecting the current adoption of LPWAN technologies and how these technologies might be incorporated in the Semantic Web. It will look in detail at different levels of interoperability in order to make these technologies future proof. In short, the main research challenges are:

1. To what extent offer current standards the possibility to connect LPWAN technologies to the Internet following the end-to-end principle?

2. Are newly developed solutions, such as SCHC, practically feasible solutions that deliver tools to keep the intelligence of the network at the edge?

3. How and to what extent can SCHC assist in the incorporation of multiple LPWAN technologies on a single constrained device? What are the necessary mechanisms to roam between heterogeneous networks to assist the low bandwidth constraints?

4. Given the low bandwidth of LPWAN technologies, how can data from these networks be enriched with semantics in order to fit the requirements of the Semantic Web?

5. How can the above solutions and other, proprietary technologies that do not support IPv6 be incorporated in the Semantic Web and other, non-interoperable platforms?

1.4 Outline

This dissertation is composed out of a number of scientific papers. The papers that are part of this book provide an overview of the work that has been done during this PhD. The research contributions are provided in Section 1.5 and a complete list of published papers can be found in Section 1.6. Except for Chapter 2, not much editing has been done to the original manuscripts. Chapter 2.3 has been extended to match the nomenclature of the current specification, Figure 2.3, 2.4, 2.5, and 2.6 were added to better assist the text. Figure 2.7 was updated in order to provide a clearer example of the fragmentation protocol. Future Work sections were removed from Chapter 2, 4, 5 and 6, since they were summarized and rewritten in Chapter 7. The remainder of this Section provides an overview of the outline of this book.

Chapter 2 serves as an introduction to the background technologies that lay the foundation of this dissertation. This Chapter introduces the wireless communication technologies and the available protocol stacks for constrained devices. An analysis of these stacks demonstrates that a novel standard is required to provide network interoperability in the area of LPWAN. The remainder of the Chapter demonstrates that this can be provided efficiently by the novel SCHC protocol. This adaptation layer, that sits between the Medium Access Control (MAC) layer and network layer, is compared to

other solutions and implemented on a constrained device in order to evaluate the software footprint. Part of this Chapter was published in early 2019. However, it was not until April 2020 that the LPWAN WG standardized the specification. Therefore, this Chapter has been updated in order to complement the original paper with a more in depth explanation about the technical aspects of the specification. Secondly, the terminology changed slightly and has been updated in order to match the nomenclature of the published standard.

As Chapter 3 will point out, LPWANs should abolish their proprietary Application Programming Interfaces (APIs) in order to truly provide network interoperability. Therefore, this Chapter analyses the shortcomings of current systems and proposes methods to decentralize LPWANs. Based on a real world example, where the IPv6 of the destination is not known before deployment, shortcomings of the specification are laid out. These requirements led to the conclusion that a management protocol is needed in these types of networks. After analysing current device management and registration mechanisms, conclusions are drawn that there is a need for a new registration and configuration protocol in SCHC networks. However, none of the existing solutions can serve as a provisioning tool in such low power networks and consequently, two provisioning protocols are proposed: the SCHC Registration mechanism and the SCHC Optimized Neighbor Discovery mechanism. Both mechanisms are evaluated in a simulation based environment, where their success rate and registration time is measured to draw conclusions about their feasibility.

The next Chapter will look at more complex settings and configurations. Since many of the aforementioned wireless technologies fail to provide the required bandwidth for e.g. OTAUs, some use cases require the combination of multiple technologies. Such cases can deploy multiple LPWAN technologies in order to cope with the limitations of one another. However, for such requirements, a new architecture is needed. This architecture is able to deliver device interoperability, by keeping track of the active network. Since devices can now switch between networks, an energy efficient network discovery algorithm is presented too, which was implemented in a real world use case and evaluated in a simulation based environment. The energy consumption for different configurations of the parameters are modeled using an energy model for three different technologies: Sigfox, Long Range Wide Area Network (LoRaWAN) and DASH-7.

Chapter 2 to 4 provide interoperable solutions to access the resources of constrained IoT devices. However, as the future Web will move towards a Semantic Web, data should be enriched with semantics in order to be understandable by machines, accessible by different platforms and to link to

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

	Ch.2	Ch.3	Ch.4	Ch.5	Ch. 6
Analysis and implementation of the SCHC protocol	•	•			
Analysis and design of SCHC context provisioning		•			
Analysis and optimization of multimodal LPWAN solutions	•		•		
Analysis of semantic interoperability in LPWANs				•	
Design and evaluation of an interoperability platform for LPWANs			•		•

other pieces of data. Therefore, Chapter 5 studies semantic interoperability in LPWANs. First, an analysis is performed to determine the requirements to incorporate constrained IoT devices into the future Web. The evaluation demonstrates that their bandwidth constraints make it infeasible to use semantic Web technologies, such as Resource Description Framework (RDF), over a constrained wireless link. Therefore, this chapter analyses the implementation of a Light Weight Machine to Machine (LwM2M) based solution, which resulted in an architecture that uses a middleware layer to convert the LwM2M data to RDF.

Both Chapter 4 and 5 demonstrate that a middleware layer is required to provide network interoperability, device interoperability and semantic interoperability. These observations are leveraged upon in the final Chapter to build a generic architecture that provides interoperability on the aforementioned levels. It is extended in such a way that it can provide platform interoperability, while maintaining syntactic interoperability. Furthermore, it is demonstrated that the generic approach allows the addition of other services, such as localization. This Chapter evaluates these concepts in the scope of the Port of the Future based on five levels of interoperability, platform scalability and performance.

Table 1.2 gives an overview of the targeted challenges per chapter. This, together with the table of contents, may aid in navigating throughout this dissertation.

1.5 Contributions

- Implementation and analysis of the SCHC protocol (Chapter 2)

  - Development of device agnostic open source SCHC library in C
    * Implementation of the compressor and decompressor for the IoT protocol stack: CoAP/LwM2M, UDP, IPv6
    * Implementation of fragmenter and reassembler state machines for the reliability modes: No-Ack, Ack-on-Error and Ack-Always
    * Implementation of low-footprint network memory buffer manager
    * Implementation of low-footprint connection manager

- Design of SCHC context provisioning protocols (Chapter 3)
  - Analysis of existing network management protocols and trade-offs in IPv6 networks
  - Design of the SCHC Registration provisioning protocol and the SCHC Optimized Neighbor Discovery provisioning protocol
    * Comparison of both mechanisms in terms of flexibility
    * Quantification of the packet overhead for both mechanisms in the Ack-On-Error reliability mode
    * Feasibility of both mechanisms in a simulation based environment in terms of energy overhead and required registration time

- Design of an energy efficient handover protocol for LPWANs (Chapter 4
  - Design and implementation of an architecture for devices in multimodal LPWAN configurations
  - Design and implementation of a low power network discovery method
  - Design of an energy model for LoRaWAN, SigFox and DASH-7
  - Assessment of the algorithm in a simulation based environment, based on the energy overhead, network discovery time and the reliability latency
  - Providing insights in the algorithm's parameter configuration trade-offs and implementation overhead

- Methods for LPWANs to support Linked Data and connect to the Semantic Web (Chapter 5)
  - Design and implementation of an architecture to connect LPWAN devices to a Linked Data Platform (LDP)

  - Providing insights in the feasibility of transmitting RDF data over a constrained link
  - Mapping of LwM2M data to RDF in order to support semantic interoperability in LPWANs

- Design and implementation of a generic architecture for multi-level interoperability in the constrained Internet of Things (Chapter 6)
  - Design and implementation of a scalable, open-source middleware platform in the scope of the Port of the Future
  - Evaluation and assessment of the shortcomings of existing IoT platforms
  - Implementation and evaluation of the platform in terms of interoperability, scalability, performance and resource consumption, compared with existing platforms

1.6 List of Publications

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

1.6.1 Publications in international journals (listed in the Science Citation Index [3])

1. **Bart Moons**, Michiel Aernouts, Vincent Bracke, Bruno Volckaert, Jeroen Hoebeke. *FLINT: Flows for the INTernet of Things* Published in MDPI Applied Sciences, 2021: 11(19). DOI: 10.3390/app11199303 **IF: 2.679**

2. Celia Garrido-Hidalgo, Jetmir Haxhibeqiri, **Bart Moons**, Jeroen Hoebeke, Teresa Olivares, F. Javier Ramirez, Antonio Fernandez-Caballero. *LoRaWAN scheduling: from concept to implementation* Published in IEEE Internet of Things Journal, 2021: 8(16). p.12919-12933. DOI: 10.1109/JIOT.2021.3064430 **IF: 9.471**

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

3. Vincent Bracke, Merlijn Sebrechts, **Bart Moons**, Jeroen Hoebeke, Filip de Turck, Bruno Volckaert. *Design and evaluation of a scalable Internet of Things backend for smart ports* Published in Software: Practice and Experience, 2021: 51(7). DOI: 10.1002/spe.2973 **IF: 2.028**

4. Michiel Aernouts, Filip Lemic, **Bart Moons**, Jeroen Famaey, Jeroen Hoebeke, Maarten Weyn, Rafael Berkvens. *A multimodal localization framework design for IoT applications* Published in MDPI Sensors, 2020: 20(16). DOI: 10.3390/s20164622 **IF: 3.576**

5. Subho Shankar Basu, Jetmir Haxhibeqiri, Mathias Baert, **Bart Moons**, Abdulkadir Karaagac, Pieter Crombez, Pieterjan Camerlynck, Jeroen Hoebeke. *An end-to-end LwM2M-based communication architecture for multimodal NB-IoT/BLE devices* Published in MDPI Sensors, 2020: 20(8). DOI: 10.3390/s20082239 **IF: 3.576**

6. **Bart Moons**, Abdulkadir Karaagac, Eli De Poorter, Jeroen Hoebeke. *Efficient vertical handover in hetreogeneous low-power wide area networks* Published in IEEE Internet of Things Journal, 2020: 7(3). p.1960-1973. DOI: 10.1109/JIOT.2019.2961950 **IF: 9.471**

1.6.2 Publications in international conferences (listed in the Science Citation Index [4])

1. Subho Shankar Basu, Jetmir Haxhibeqiri, **Bart Moons**, Jeroen Hoebeke. *An energy-efficient multi-modal IoT system leveraging NB-IoT and BLE.* Published in the proceedings of the 2020 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS), 27-28 January 2021. p. 31-37. Online, Bali, Indonesia.

2. **Bart Moons**, Jetmir Haxhibeqiri, Abdulkadir Karaagac, Eli De Poorter, Jeroen Hoebeke. *Using SCHC for an optimized protocol stack in multimodal LPWAN solutions.* Published in the proceedings of the 5th IEEE World Forum on Internet of Things (WF-IoT), 15-18 April 2019. p. 430-435. Limerick, Ireland.

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

3. Abdulkadir Karaagac, Matthias Van Eeghem, Jen Rossey, **Bart Moons**, Eli De Poorter, Jeroen Hoebeke. *Extensions to LwM2M for intermittent connectivity and imporved efficiency.* Published in the proceedings of the 4th IEEE Conference on Standards for Communications and Networking (IEEE CSCN), 29-31 October 2018. p. 1-6. Paris, France.

4. Jeroen Hoebeke, Jetmir Haxhibeqiri, **Bart Moons**, Matthias Van Eeghem, Jen Rossey, Abdulkadir Karaagac, Jeroen Famaey. *A cloud-based virtual network operator for managing multimodal LPWAN networks and devices.* Published in the proceedings of the 3rd Cloudification of the Internet of Things Conference (CIoT), 2-4 July 2018. p. 1-8. Paris, France.

5. **Bart Moons**, Jetmir Haxhibeqiri, Matthias Van Eeghem, Jen Rossey, Abdulkadir Karaagac, Stefano Quattrocchi, Jeroen Famaey, Jeroen Hoebeke. *DEMO: A cloud-based virtual network operator for managing multimodal LPWAN networks and devices.* Published in the proceedings of the 3rd Cloudification of the Internet of Things Conference (CIoT), 2-4 July 2018. p. 1-2. Paris, France.

1.6.3 Publications in other International Conferences

1. **Bart Moons**, Jeroen Hoebeke. *Towards an Open Web of Things.* Published in the proceedings of the 2020 IEEE International Symposium on Technology and Society (ISTAS), 12-15 November 2020. p. 176-179. Online.

2. **Bart Moons**, Flor Sanders, Thijs Paelman, Jeroen Hoebeke. *Decentralized linked open data in constrained wireless sensor networks* Published in the proceedings of the 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS), 14 December 2020. Online.

1.6.4 Publications in other International Journals

1. **Bart Moons**, Eli De Poorter, Jeroen Hoebeke. *Device discovery and context registration in static context header compression networks* Published in MDPI Information, 2021: 12(2). p.1960-1973. DOI: 10.3390/info12020083.

References

[1] G. R. Gromov. *The Roads and Crossroads of Internet History*. Online Writings, 1995. Available from: http://www.netvalley.com/intval.html.

[2] B. Elmore. *January 2017: From the Transatlantic Telephone to the iPhone*. Origins, January 2017. Available from: https://origins.osu.edu/milestones/january-2017-transatlantic-telephone-iphone.

[3] S. Crocker, J. Postel, J. Newkirk, and M. Kraley. *An Official Protocol Proffering*. Technical Report RFC57, RFC Editor, June 1970. Available from: https://datatracker.ietf.org/doc/html/rfc54.

[4] R. Kalin. *A Simplified NCP Protocol*. Technical Report RFC60, RFC Editor, July 1970. Available from: https://datatracker.ietf.org/doc/html/rfc60.

[5] P. Kirstein. *Early experiences with the Arpanet and Internet in the United Kingdom*. IEEE Annals of the History of Computing, 21(1):38–44, January 1999. Conference Name: IEEE Annals of the History of Computing. doi:10.1109/85.759368.

[6] V. G. Cerf and R. E. Kahn. *A Protocol for Packet Network Intercommunication*. IEEE Transactions on Communications, 22(5):637–648, 1974. doi:10.1109/TCOM.1974.1092259.

[7] V. Cerf, Y. Dalal, and C. Sunshine. *Specification of Internet Transmission Control Program*. Technical Report RFC675, RFC Editor, December 1974. Available from: https://datatracker.ietf.org/doc/html/rfc675.

[8] J. Postel. *Transmission Control Protocol*. Technical Report RFC793, RFC Editor, September 1981. Available from: https://datatracker.ietf.org/doc/html/rfc793.

[9] J. Postel. *Internet Protocol*. Technical Report RFC791, RFC Editor, September 1981. Available from: https://datatracker.ietf.org/doc/html/rfc793.

[10] T. Berners-Lee. *Information Management: A Proposal*. CERN, page 20, 1989.

[11] D. Evans. *How the Next Evolution of the Internet Is Changing Everything*. Cisco Internet Business Solutions Group, page 11, 2011. Available from: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[12] *Amendment of the rules to authorize spread spectrum and other wideband emissions in the Public Safety and Industrial, Scientific, Medical-Bands.*, February 2016. Available from: https://www.fcc.gov/document/amendment-rules-authorize-spread-spectrum-and-other-wideband.

[13] U. Raza, P. Kulkarni, and M. Sooriyabandara. *Low Power Wide Area Networks: An Overview.* arXiv:1606.07360 [cs], June 2016. arXiv: 1606.07360. Available from: http://arxiv.org/abs/1606.07360.

[14] P. Eugenio. *5 things to know about the LPWAN market in 2020*, January 2020. Available from: https://iot-analytics.com/5-things-to-know-about-the-lpwan-market-in-2020/.

[15] A. Strategy. *Strategy Analytics: Unlicensed LPWA IoT Connections Will Grow to 400 Million By 2025*, December 2019. Available from: https://www.businesswire.com/news/home/20191219005087/en/Strategy-Analytics-Unlicensed-LPWA-IoT-Connections-Will-Grow-to-400-Million-By-2025.

[16] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J.-C. Zúñiga. *SCHC: Generic Framework for Static Context Header Compression and Fragmentation.* Request for Comments RFC 8724, Internet Engineering Task Force, April 2020. Num Pages: 71. Available from: https://datatracker.ietf.org/doc/rfc8724, doi:10.17487/RFC8724.

[17] D. Guinard, V. Trifa, F. Mattern, and E. Wilde. *From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices.* In D. Uckelmann, M. Harrison, and F. Michahelles, editors, Architecting the Internet of Things, pages 97–129. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. Available from: http://link.springer.com/10.1007/978-3-642-19157-2_5.

[18] J. Miranda, N. Mäkitalo, J. Garcia-Alonso, J. Berrocal, T. Mikkonen, C. Canal, and J. M. Murillo. *From the Internet of Things to the Internet of People.* IEEE Internet Computing, 19(2):40–47, March 2015. Conference Name: IEEE Internet Computing. doi:10.1109/MIC.2015.24.

2

Overview and implementation of the SCHC standard

This chapter gives an overview of the available standards and solutions that can be used to provide network interoperability in Low Power Wide Area Network (LPWAN) solutions. It presents a comparative study of the available standards and an analysis of the Static Context Header Compression (SCHC) standard as a common layer on top of a combination of three different long range technologies. As will be shown, some LPWAN technologies do not support the use of traditional protocols and adaptation layers. Therefore, this Chapter presents an implementation and analysis of SCHC, which was developed to fill this gap, and served as a basis for this dissertation.

⋆⋆⋆

Several problems arise when introducing the many embedded devices the Internet of Things (IoT) is expected to bring. To move away from complex gateways that translate between proprietary protocols and standardized internet languages, a shift to internet standards is needed to address the huge amount of devices. From an addressing point of view, the Internet Protocol Version 6 (IPv6) protocol seems to suit this need. Combined with other standardized protocols like the Constrained Application Protocol (CoAP) and User Datagram Protocol (UDP), it may become one of the preferred ways to communicate with constrained devices. However, considering the severe

energy, memory, processing and communication constraints of LPWAN devices and networks, this stack is not a viable solution. The overhead brought by its headers (i.e. IPv6: 40 bytes, UDP: 8 bytes, CoAP: typically around 20 bytes) is not in line with the few tens to hundreds of bytes which can be transmitted over these low power communication technologies. LPWANs are often limited by duty cycle regulations, as they operate in unlicensed spectrum, which may limit the time on air and, consequently, the size of packet transmissions when operating at low bit rates.

Apart from that, more demanding and diverse IoT applications could require the devices to be equipped with multiple technologies. These devices could benefit from long range outdoor communication combined with higher data rate indoor communication. Also, the fact that LPWAN equipment may be shared across different organizations makes the concept of multimodal LPWAN solutions an interesting research topic. Moreover, when using a single application across different technologies, a single common stack is desirable. In order to pick the right stack, this Chapter evaluates different protocol stacks for use in multimodal LPWAN solutions, considering Long Range Wide Area Network (LoRaWAN), DASH-7 and SigFox.

This chapter is based on the following publications:

- Bart Moons, Jetmir Haxhibeqiri, Abdulkadir Karaagac, Eli De Poorter, Jeroen Hoebeke. *Using SCHC for an optimized protocol stack in multimodal LPWAN solutions.* Published in the proceedings of the 5th IEEE World Forum on Internet of Things (WF-IoT), 15-18 April 2019. p. 430-435. Limerick, Ireland.
- Bart Moons, Abdulkadir Karaagac, Eli De Poorter, Jeroen Hoebeke. *Efficient vertical handover in hetreogeneous low-power wide area networks* Published in IEEE Internet of Things Journal, 2020: 7(3). p.1960-1973. DOI: 10.1109/JIOT.2019.2961950

2.1 Low Power Wide Area Networks

Low Power Wide Area Networks (LPWANs) are formed out of cheap sensors running applications that require low bandwidth communications over a long range. Currently, several technologies are emerging in this domain, providing low cost and low power by using mainly bands in the sub-Gigahertz (GHz) spectrum.

Depending on the region, certain restrictions may apply in order to share the spectrum. For example, in the European 868 Mega Hertz (MHz) band,

duty cycle restrictions apply between 0.1% and 10% over a period of 60 minutes, where the highest throughput channel is often used for downlink communication. Also in India do duty cycle restrictions apply to the entire band [1]. Contrarily, in the United States (US), a duty cycle of 2 to 4% applies over a period of 20 to 10 seconds to the channel and bandwidth that are being used in the 915 MHz band. Similar regulations apply in Brazil and Canada. Finally, parameters for polite spectrum access can be dictated by governments. In Japan, a minimum time window is given during which the sensor device must listen to detect if other devices are currently transmitting.

2.1.1 LoRa

Long Range (LoRa) is a Radio Access Technology in the unlicensed sub-GHz band using Chirp Spread Spectrum (CSS) modulation, patented by Semtech in 2014. CSS spreads out a narrow band signal over a wider channel bandwidth, making it more robust to noise and interference. Multiple Spreading Factors (SFs) are supported by LoRa, i.e. SF7 - SF12, offering a trade-off between a higher data rate and a longer range respectively [2]. By using Forward Error Correction (FEC) with Coding Rates (CRs) ranging from 4/5 to 4/8, even more robustness can be provided [3]. The data rate and range are affected by a combination of SF, CR and chosen bandwidth.

On top of the LoRa physical layer, the open LoRaWAN Medium Access Control (MAC) layer standard has been defined by the LoRa Alliance. LoRaWAN also defines how the network operates and how devices communicate with each other. This layer provides a medium access control mechanism and defines three types of end-devices: Classes A - C, mainly providing different ways of bidirectional communication. For a LoRaWAN Class A device, each uplink transmission is followed by two downlink receive windows of 1 and 2 seconds respectively, during which the end-device will listen for a preamble, indicating downlink communication. Class B devices will listen for downlink traffic at predefined times after synchronization to the network server using network beacons. Devices continuously listening for downlink packets are of type Class C.

Each LoRa packet starts with a programmable preamble part, ranging from 6 to 65.532 symbols, followed by 2 sync words and 2 downchirp symbols, used to synchronize traffic between sender and receiver. Therefore, header overhead of the physical layer can be very limited. When the length of the payload is known in advance and configured on both sides, the Extended Header (EH) can also be removed, which otherwise contains the payload length and a Cyclic Redundancy Check (CRC) encoded with a code rate of 4/8 [4].

2.1.2 DASH-7

The DASH-7 Alliance (D7A) specifies a full vertical network stack, covering the complete Open Systems Interconnection (OSI) model, focusing on mid-range communication. It was initially developed for 433 MHz wireless communication, based on the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 18000-7 standard, which now also includes the 868 MHz Short Range Device (SRD) and 915 MHz Industrial, Scientific and Medical (ISM) bands [5, 6]. The presentation layer contains D7AP files, consisting of configurations and user data, which can be executed as scripts. DASH-7 applications are intended to be built using those files. The downside of this full vertical stack is that it introduces a lot of overhead when using an IPv6 based standardized approach. In fact, when using the DASH-7 specification, a CoAP/UDP/IPv6 packet must be encapsulated on top of the DASH-7 application layer protocol.

In a DASH-7 network two types of communication models can be used. The first one is based on low power wake up, where a sleeping end device will discover a requesting signal by regularly waking up and detecting an advertising frame containing the time of the upcoming request, as shown in Figure 2.1.

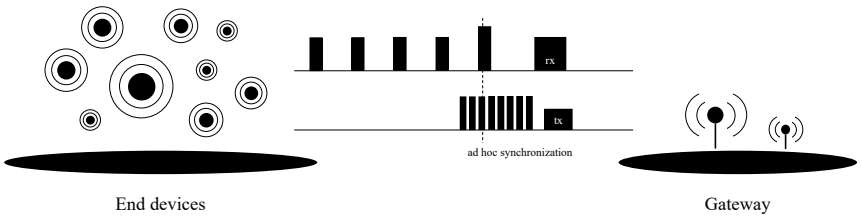


Figure 2.1: DASH-7 low power wake up and ad hoc synchronization. Sleeping end-points can detect incoming requests by listening to a sync train that contains the time of the upcoming request.

The second approach, i.e. Dormant Sessions, is used in a push-based model where sensor devices send unsolicited requests to the gateway when a (pre-)configured criteria is met. Using this model, it is not possible for the device to receive requests from the gateway to update e.g. a configuration. In this case, the gateway can queue a command and wait for a message from the device within a certain timeout. If the addressed endpoint happens to start a dialog with the gateway before the timeout has elapsed, the gateway can return a flag in its response to the endpoint, signalling a pending request. From this moment on the dialog is extended and the gateway becomes the requester and the endpoint the responder. The request is then transmitted

as part of this extended dialog. If a sensor node does not send a notification within this timeout, the gateway will try to wake up the sensor device using the low power wake up mechanism with a message saying that a query is ready.

Data rates, modulation types, pass-and stopband requirements are specified by Channel Classes. Data is encoded using PN9 encoding, which might be used in combination with a FEC scheme. As the FEC encoder is a 1/2 rate convolutional code, the data rate will decrease accordingly. Each packet is preceded by a ramp-up period and a preamble to synchronize the clock of the receiver followed by a sync word to align the packet payload. As all these parameters are configurable, more or less robustness, and consequently overhead, can be provided [6].

2.1.3 Sigfox

Sigfox is a proprietary technology, patented by the French company of the same name. Sigfox itself or in partnership with others offer an already deployed end-to-end 0G network, which already covers more than 60 countries. Their technology is based on Binary Phase Shift Keying (BPSK) in the 868 and 915 MHz ultra narrow SRD band (depending on the region) [2]. Narrowband modulation techniques are able to obtain a higher link budget because the noise level in a single narrow band is minimal [7]. The benefit of such a robust radio signal decreases the data rate, whereas the time-on-air increases. Another drawback of the Ultra Narrowband (UNB) modulation is the significant link asymmetry. Downlink communication is limited to four 8-byte messages every day.

2.1.4 Others

The previous paragraphs aim to provide information about technologies that will be used throughout this dissertation. However, for completeness, other relevant technologies are discussed below.

- Cellular technologies: Narrowband IoT (NB-IoT) and Long Term Evolution (LTE)-Machine Type Communication (MTC) (LTE-M) are both cellular technologies proposed by the 3rd Generation Partnership Project (3GPP). Both aiming to provide low power, low cost and long range with improved in- and outdoor coverage. LTE-M provides low latency and higher bandwidth, whereas NB-IoT targets devices with ultra-low device cost and power consumption, at the expense of up to 10 seconds latency. The most important distinction between cellular and other LPWAN technologies is their operation in the licensed spectrum

Table 2.1: Differences between Weightless-W/-N/-P

	Weightless-N	Weightless-P	Weightless-W
Directionality	1-way	2-way	2-way
Feature set	Simple	Full	Extensive
Range	5km+	2km+	5km+
Battery life	10 years	3-8 years	3-5 years
Feature set	Very low	Low	Low-medium

and hence, their ability to communicate without a duty cycle limitation [8]. Although NB-IoT reuses many of the mechanisms defined in LTE, two extensions have been defined in order to save power. The first one being the extended Discontinuous Reception (DRX) (eDRX), which is used to check the paging channel periodically for incoming data and has been extended from 2.56s in LTE to 175 minutes in NB-IoT. Even more power can be saved by entering Power Saving Mode (PSM), which allows the constrained device to remain registered to the network, without monitoring the paging channel. The duration of the PSM cycle can last up to approximately 413 days. Downlink traffic is therefore limited by the periodicity of the PSM or DRX cycle [9].

- Weightless: three different standards have been proposed by the Weightless Special Interest Group: Weightless-W, deployable in TV whitespace, Weightless-P, providing high performance and the uplink-only Weightless-N protocol, focusing on ultra-low cost. The main differences are listed in Table 2.1

  In Weightless-P, every channel comprises 12.5 kHz of the spectrum and can be assigned an adaptive data rate, ranging from 200 bps to 100 kbps, organized by the time-synchronized gateways. Weightless-P is a downlink oriented protocol, whereas Weightless-N only allows uplink traffic. On the other hand, Weightless-W must be deployed in the unused spectrum of TV transmissions (400 - 800 MHz) and must adhere to the local regulations. Weightless-W spreads out its signal with variable factors from 1 to 1024. Increasing the spreading factor will increase the range, without increasing output power, however, at the cost of data throughput.

2.2 IETF Protocols for Constrained Devices

This Section discusses other Internet Engineering Task Force (IETF) initiatives that have tried to bring interoperability and IPv6 connectivity among constrained devices.

2.2.1 CoAP

The Constrained Application Protocol (CoAP) can be seen as the Hyper Text Transfer Protocol (HTTP) for constrained devices, as it uses a stateless client-server architecture. The light weight nature of CoAP makes it the perfect candidate for a LPWAN protocol stack. Every object contains a list of resources, representing data available from sensors or actions available to actuators. Every resource is accessible through a Uniform Resource Identifier (URI) and can be interacted with using protocol specific methods such as GET, PUT, POST and DELETE [10].

The original Request For Comments (RFC) was updated after a few years and introduced the FETCH, PATCH and iPATCH methods, since the existing methods only allowed access to complete resources and not parts of a resource. Compared to the non-safe, non-idempotent PATCH method, the iPATCH method is a CoAP specific method that is not safe, but is idempotent.

Four HTTP methods were not added to the CoAP specification.

- CONNECT is not yet supported in CoAP. Since CoAP works over UDP and Transport Layer Security (TLS) to Datagram Transport Layer Security (DTLS) tunneling has not yet been specified, an HTTP to CoAP proxy cannot satisfy this.

- the OPTIONS method, which requests the permitted communication for the target resource in HTTP is not supported in CoAP. This is very similar to the built-in resource discovery in CoAP and as such did not require such method.

- the HTTP HEAD method is not supported in CoAP. However, an HTTP-CoAP proxy might perform a CoAP GET request and respond with the HTTP headers and without a message-body.

- HTTP TRACE is not supported; since CoAP end points might be asleep, the TRACE method is not a reliable way to test if a server is up and supports the requested resource.

In LPWAN use cases, many sensor nodes will send frequent updates about their usage and acknowledging every message will add a significant

cumulative load on the network and is no necessity. Therefore, the following CoAP mechanisms were considered in this dissertation.

2.2.1.1 Observe

The CoAP Observe extension is a simple mechanism to retrieve a representation of a resource and keep this updated by the subject as long as the observer is interested. The extension uses a best-effort approach for sending updates to the observer.

In a CoAP Observe scenario, the data collection is always initiated by the observer, who also has to maintain all relationships for each subject. This requires a lot of bookkeeping and the back-end to know all its data sources beforehand [11].

2.2.1.2 No-Response Option

While CoAP implements a Non-Confirmable (NON) mechanism to omit the acknowledgment of a particular message, the server will still reply with a response code, due to the request/response nature of the protocol. Therefore, the working group published an amendment to the protocol specification introducing the No-Response option in order to get rid of any kind of reverse traffic [12].

In many LPWAN use cases, where downlink traffic is scarce, the No-Response option makes classic updates consume even less resources than Observe and was therefore considered a better fit. A typical CoAP request from the application can be seen in Figure 2.2. Note that in this example sensor devices are clients that have a dedicated resource on a CoAP server.

2.2.1.3 CoAP block-wise transfer

CoAP block-wise transfer can be used on top of IPv6 over Low-Power Wireless Personal Area Networks (LoWPANs) (6LoWPAN) in order to avoid 6LoWPAN fragmentation and to provide reliability. As the transfer of one fragment corresponds to a normal CoAP request, each block must be acknowledged. The failure of a single request will therefore trigger the retransmission of a single block. Each block can only have a size that is a multiple of 16 bytes. The device must ensure to use a block size smaller than the underlying layer 2 Protocol Data Unit (PDU) [13].

2.2.2 6LoWPAN and 6lo

In late 2004, the 6LoWPAN Working Group (WG) was formed to enable IPv6 over IEEE 802.15.4 networks. A few years later, the 6lo working group

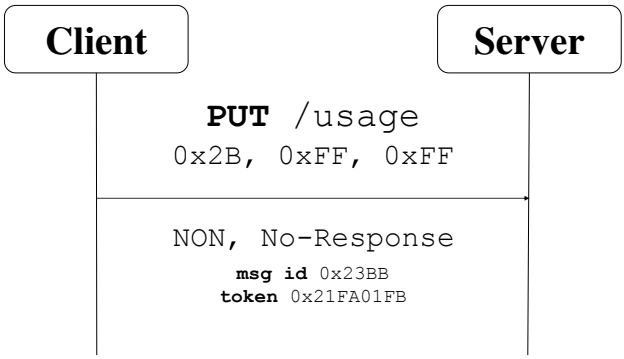


Figure 2.2: A typical No-Response LPWAN CoAP request from the application. Note that No-Response sensor devices are typically clients that have a dedicated resource on a CoAP server.

was added to bring IPv6 connectivity over several other technologies, such as Bluetooth Low Energy (BLE). [14] The IEEE 802.15.4 standard defines the maximum layer 2 PDU to be 127 bytes. 25 bytes MAC header, 40 bytes IPv6 header and 8 bytes UDP header would only leave 54 bytes for the payload, without security headers. Therefore the 6LoWPAN working group defined `LOWPAN_HC1` and `LOWPAN_HC2` in order to compress the internet protocol header and the transport layer respectively. However, these compression schemes can only be applied to link local addresses. As a consequence, the working group published an amendment, developed under the name `LOWPAN_IPHC` and `LOWPAN_NHC`. Communication using UDP and global IPv6 addresses now impacts the link layer frame with a minimum of 10 bytes overhead [15]. In order to support the IPv6 Maximum Transfer Unit (MTU) requirement of 1280 bytes, the 6LoWPAN adaptation layer also defines a simple fragmentation mechanism, which does not provide any reliability.

Furthermore, new IETF initiatives arose in order to compress a multiple of protocols using Pages and a Paging Dispatch to switch contexts. [16] In addition to this, the `6LoRH` compression technique has been developed to compress IPv6 routing information. [17]

2.3 Static Context Header Compression

Even after the above standardization efforts, some technologies still do not support the use of (adapted) internet protocols and may not benefit from

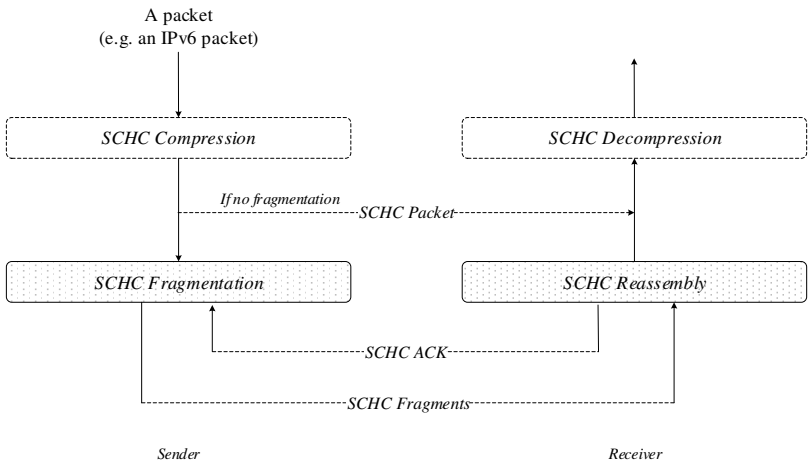


Figure 2.3: Illustration of the SCHC operations on both the sending and receiving end. If the compressed packet size still exceeds the MTU, the packet will be fragmented.

the advantages introduced by a standardized communication approach. As a consequence, the LPWAN IETF working group was formed and published the Static Context Header Compression (SCHC) mechanism as a new standard [18]. This protocol makes use of a shared, static context between two communication endpoints that contains regularly used protocol headers. The selected context is represented by an identifier in order to inform the other side about the original headers of the packet. As will be demonstrated in this Chapter, it is possible to shrink the headers down to 95% of the original size using this technique. Furthermore, the specification prescribes how large packets that don't fit in a single Layer 2 PDU should be fragmented. An overview of the SCHC operations that will be further discussed in the following subsections is given in Figure 2.3.

2.3.1 Compression

In order to perform compression, both sides of the LPWAN network share the same context, i.e. a set of rules to compress or decompress the headers. The context consists of one or more rules that are distinguished by a unique identifier. A rule describes the compression/decompression behavior for each header field and what possible compression residue (i.e. a part of the

header field that could not be compressed) to send. Each header field is matched against corresponding entries in a rule (such as the one in Table 2.2). The Fields in the SCHC context appear in the same order as in the header they represent. Every Field is labeled using a protocol parser and points to a header field of a particular protocol header. The Field Length (FL) indicates the amount of bits that are used to represent the header field. Some protocols have variable length fields, which must be indicated using a special value. Next, the Field Position (FP) is used to distinguish between fields that are used multiple times, such as the CoAP URI Path Option. The request/response nature of CoAP also requires the use of a Direction (DI) indicator, so that a rule can be used for both requests and responses. The Matching Operator (MO) is used to compare the original header field with the Target Value (TV) in the rule and can take one of the following values:

- `equal` looks for an exact match of the field value in the packet and the TV
- `ignore` ignores the field, the result is always True
- `MSB(x)` compares the first `x` bits of the packet header field and the TV
- `match-mapping` compares the packet header to a list of entries. The matching value can by identified by an index

When performing compression, every header value is matched against the corresponding rule field of every rule in the context. Once all of these Field Descriptors have an exact match with the original header that rule is selected for compression. Next, the Compression Decompression Action (CDA) can be used for every field to build the compressed header and can take one of the following actions:

- `not-sent` will not add the field to the compressed header
- `value-sent` will send the original header value to the other side
- `mapping-sent` will add the index of the matched value
- `LSB` will send the `x` last bits from the original value
- `compute-*` can be used to calculate, for example, the length or checksum
- `DevIID` can be used to build the device layer 2 address
- `AppIID` can be used to build another layer 2 address required by the technology

Table 2.2: SCHC rule used to compress the request from Figure 2.2

Field	FL	FP	DI	Target Value	MO	CDA
Version	2	1	BI	0x01	&equal	not-sent
Type	2	1	BI	0x01 (NON)	&equal	not-sent
Token Length	4	1	BI	0x04	&equal	not-sent
Code	8	1	UP	0x03 (PUT)	&equal	not-sent
Message ID	16	1	BI	0x0000	&ignore	not-sent
Token	32	1	BI	0x21FA01F0	&MSB(28)	LSB(4)
URI-Path	40	1	UP	‘‘usage"	&equal	not-sent
No-Response	8	1	UP	0x1A	&equal	not-sent

The rule ID and possibly compressed values (i.e. residue) are send to the other side of the network. The decompressor will use the Target Value and the residue to reconstruct the original value.

Table 2.2 shows an example SCHC context that can be used to compress typical requests from LPWAN applications, such as the one from Figure 2.2. Since SCHC is responsible for reliability, the application does not require acknowledgments from the CoAP layer. Therefore, the message ID may be ignored. Nevertheless, CoAP may require the use of separate tokens to intertwine a number of packet exchanges, which explains the use of the Most Significant Bit (MSB) MO on this header field. The MO will only match the most significant bits to the original value as indicated in the field length (28 in this case). The least significant bits are added to the compressed header. Our example shows a variation on the last 4 bits of the token, allowing 16 simultaneous requests and responses.

Listing 2.1 shows that after adding the rule ID, a matching rule will compress the 18 bytes long CoAP header down to 1 byte (4 bits Token + 4 bits padding), followed by the application data. IPv6 and UDP headers can be compressed in the same way.

Listing 2.1: extract of the CoAP layer compression

```
[001] network_layer_send(): sending 22 bytes of
   application layer data
[002] CoAP header (18 B):
[003] 54 03 23 BB 21 FA 01 FB B5 75 73 61 67 65 D1
   EA 1A FF
[004] Application data (4 B):
```

Figure 2.4: Detailed header format for the All-1 SCHC Fragment.

```
[005] 00 00 00 07
[006] schc_compress(): compressing 18 bytes of CoAP
    header
[007] Rule id: 21 (0x15)
[008] 15 B0 00 00 00 70
```

2.3.2 Fragmentation

Once a packet has been compressed or remained uncompressed, the size of the resulting SCHC packet is matched to the underlying layer 2 MTU. A packet exceeding the L2 data unit, may be fragmented by the SCHC Fragmenter. This operation will chop the SCHC packet into fragments, called tiles. A number of tiles can belong to a window with a pre-defined size. All tiles belonging to the same window carry the same window bit. The reception of a window may or may not be acknowledged, depending on the reliability mode. The order of fragments is indicated with the Fragment Count Number (FCN). All FCN bits set to 1, called an all-1 window, indicates the last fragment of a packet. Such tile contains a Reassembly Check Sequence (RCS) in order to validate the integrity of the message. This is computed over the original, compressed SCHC packet that has been concatenated with possible padding bits. The header format is given in Figure 2.4.

A tile with an FCN set to 0, on the contrary, is called an All-0 window and denotes the last fragment of a window. These and all other tiles have the same structure as the All-1 tile, except for the RCS field. Every tile carries a Rule ID to identify the fragmentation parameters and can be used simultaneously by multiple SCHC packets. In order to differentiate between these packets, a Datagram Tag (DTag) can be used. Each window consists of a number of fragments, defined by `N`, which will trigger the receiver to send an acknowledgment. For some technologies, such as SigFox, it will be more interesting to use a larger window size, since only a limited number of downlink messages are possible each day. An overview of the fragmentation packetization is given in Figure 2.5.

Once a complete window is received, one of the three reliability modes can be used to ensure a correct reassembly and to offer optional reliability:

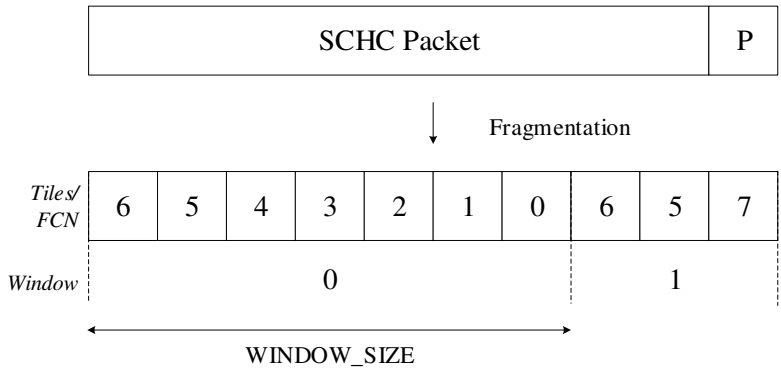


Figure 2.5: Overview of the SCHC fragmentation packetization. A SCHC packet is chopped in 10 fragments, with a WINDOW_SIZE equal to 7. The Window number and FCN is given for each tile.

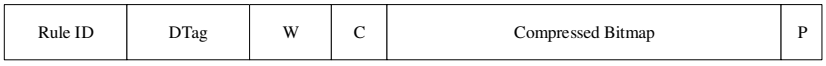


Figure 2.6: Format of the SCHC ACK Message. If C is set to 1, no compressed bitmap is sent.

- ACK-Always: each window is acknowledged, regardless of any missing fragments
- ACK-on-Error: only windows are acknowledged when a fragment belonging to that window was lost
- No-ACK: no reliability is offered beyond that of the underlying communication technology

An ACK consists of the rule ID, the DTag, the window number and the encoded bitmap. The sender will check the window bit to verify that the ACK belongs to the correct window. The bitmap that is included in the ACK indicates whether any fragments have been lost or not by toggling a bit at the position corresponding to the tile number of that window. The header format is given in Figure 2.6.

Figure 2.7 illustrates how a packet is sent using reliability mode Ack-Always. Since WINDOW_SIZE equals 7, a window can consist of up to 7 fragments. Therefore, the original SCHC packet is chopped into 10 fragments.

The example illustrates that during the transmission of the first window, the 4th (with FCN 3) and last (FCN 0) fragments are lost. Since the last fragment is used to trigger an ACK from the receiver, no acknowledgement will be sent by the receiver. Therefore, the sender sets a retransmission timer to identify any lost all-x fragments or acknowledgments. In order to request an acknowledgement, the sender will send an `SCHC ACK REQ` fragment after an invocation of the retransmission timer. This message format is used to trigger the receiving side to resend an acknowledgment. The receiver answers the `SCHC ACK REQ` with an acknowledgement that contains the bitmap. Every bit that has been set to zero indicates a missing fragment which must be retransmitted in the next retransmission phase.

In the example, the retransmission is acknowledged with a full bitmap. This triggers the sender to move to the next window, which uses the same transmission method. However, since this is the last window, the final fragment, the *all-1 fragment* with FCN 7, will trigger the receiving end to calculate the RCS. Since the fragment with FCN 5 was lost, the final ACK has the C bit set to 0, meaning that the RCS, over the entire SCHC packet, was unsuccessful. Therefore, the receiving side will request any lost fragments by sending the bitmap to the sender. The sender will again reply with the requested fragments that must be acknowledged by the receiver. In the final stage of the example, it is shown how a lost ACK can be detected by the receiver using a timeout and can be requested again using the `SCHC ACK REQ` format.

2.4 Multi protocol analysis

Choosing the correct protocol for an LPWAN application is not an easy task. Therefore, before having made the decision to continue with SCHC for any of the selected LPWAN networks, we have evaluated the different protocols described in section 2.2 for a device which is able to switch between multiple technologies: Sigfox, LoRaWAN and DASH-7. Sigfox may be used for cross-country tracking, while LoRaWAN or DASH-7 may be used when a higher throughput is required.

Figure 2.8 shows how every protocol will handle header compression, packet fragmentation and packet loss. In this example we consider the best case only for the different protocol stacks. This means that the IPv6 headers may be compressed down to 10 bytes using `LOWPAN_IPHC` and `LOWPAN_NHC` for 6LoWPAN, the CoAP header is limited to 7 bytes, since the CoAP client issues a simple GET request to the shortest URI possible (`/1`), making use of a token with length 2.

We will consider the following scenarios in our analysis:

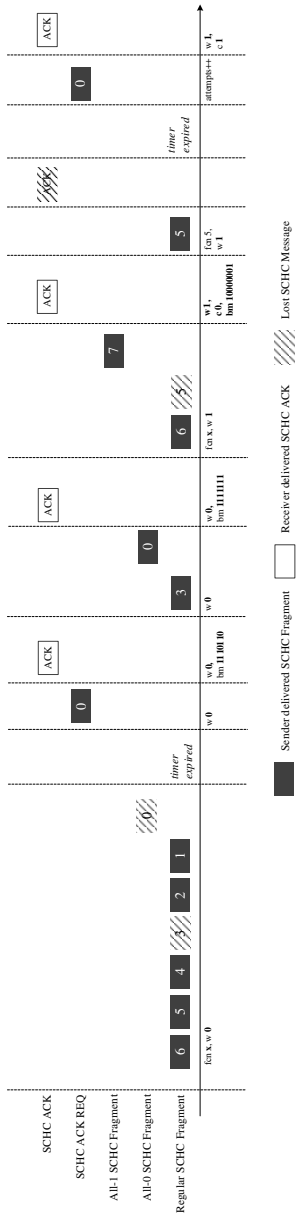


Figure 2.7: Illustration of the Ack-Always reliability mode; every window must be acknowledged. Missing all-0 fragments are detected using timers. Time moves rightward.

Table 2.3: Header overhead and number of exchanged packets for different MTUs and 128 bytes of payload

	6LoWPAN		CoAP block-wise		SCHC	
	(0)	(1)	(0)	(1)	(0)	(1)
MTU 12 bytes						
packets	X	X	X	X	16	18
overhead UL	X	X	X	X	7 B	10 B
overhead DL	X	X	X	X	24 B	25 B
MTU 51 bytes						
packets	5	10	8	10	5	7
overhead UL	36 B	72 B	72 B	90 B	9 B	10 B
overhead DL	23 B	46 B	95 B	115 B	4 B	7 B
MTU 242 bytes						
packets	2	4	4	6	2	4
overhead UL	23 B	47 B	47 B	54 B	1 B	2 B
overhead DL	17 B	34 B	36 B	71 B	1 B	2 B

1. CoAP with 6LoWPAN compression and fragmentation
2. CoAP blockwise transfer and 6LoWPAN compression
3. SCHC compressed CoAP, IPv6 and UDP with SCHC fragmentation

In order to achieve reliable communication in all three cases, a confirmable CoAP message is sent in the first case, since no reliability is guaranteed by the lower layers. CoAP block-wise transfer will confirm each block for the second case and as for the SCHC case, the ACK-Always reliability mode is used.

Table 2.3 compares the behavior of the different protocols over different technologies. The first column `(0)` represents the case where no packets were lost, the second `(1)` where 1 packet was lost.

As the 6LoWPAN fragment offset can only express a multiple of eight bytes [19], the first packet minimally contains 12 bytes, i.e. 4 bytes header and 8 bytes payload. All consecutive packets carry 5 bytes header and therefore contain at least 13 bytes. Since SigFox has a MTU of 12 bytes and LoRaWAN a global minimum of 11 bytes, 6LoWPAN fragmentation is not suitable for this technology. CoAP block-wise transfer could be used to prevent 6LoWPAN fragmentation. Nevertheless, the smallest block size CoAP supports is 16 bytes and will therefore not be transferable over Sigfox. SCHC only requires 12 bits header, which leaves room for another

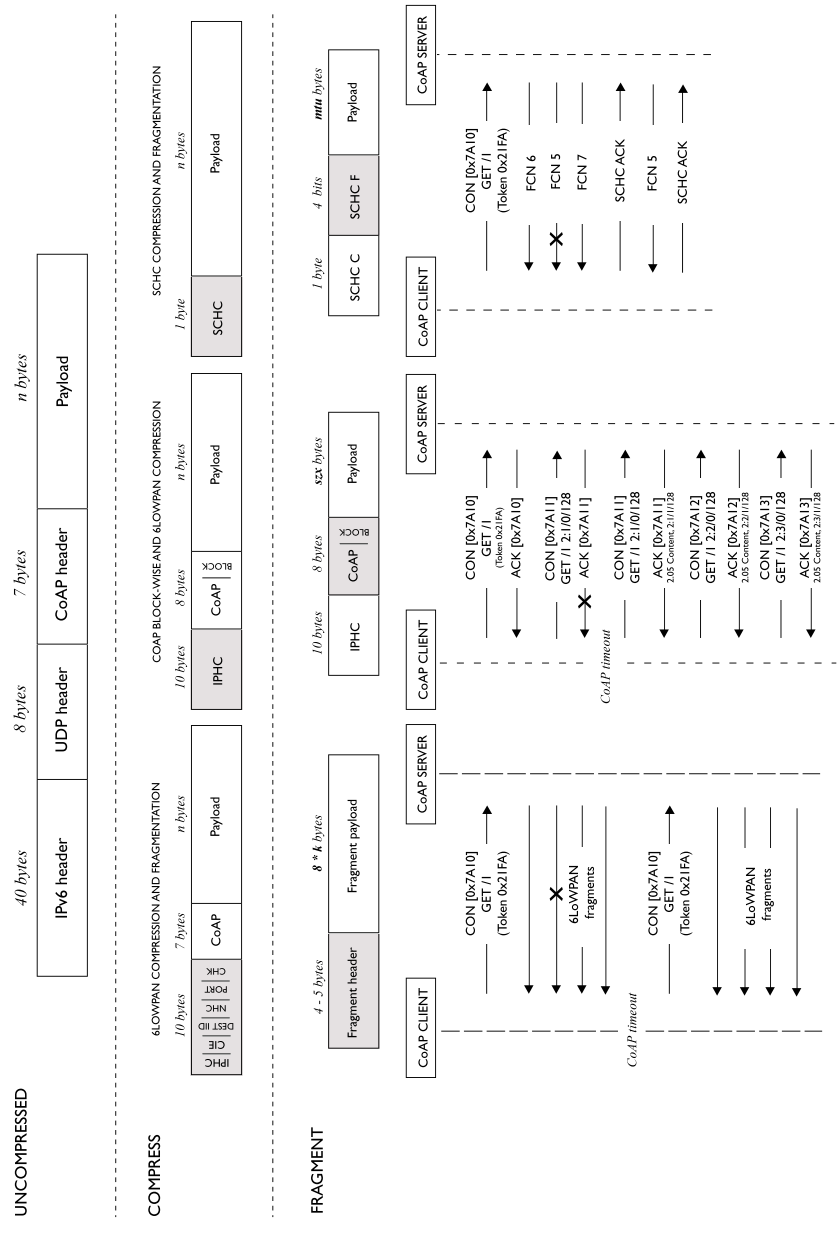


Figure 2.8: Multi protocol analysis to enable IPv6 end-to-end connectivity on a LPWAN device. The flow diagram supposes a MTU of 51 bytes and application payload of 128 bytes.

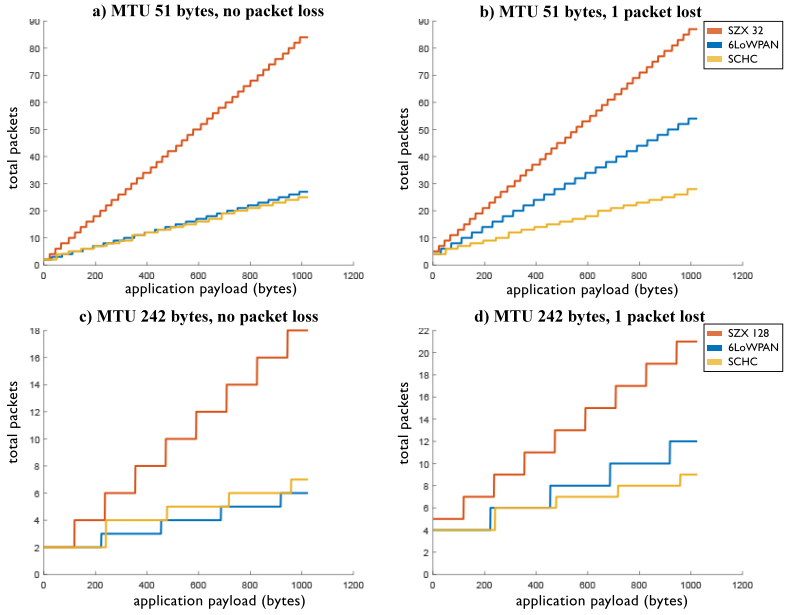


Figure 2.9: Number of exchanged packets for different payload sizes and MTUs

84 bits of payload and seems the only suitable protocol which supports IPv6 connectivity for all LPWAN technologies.

An MTU of 242 bytes will leave the packet unfragmented for the SCHC and the 6LoWPAN case. We suppose that no fragmentation means that the packet over SCHC gets acknowledged when using a CoAP CON request.

From this table we can conclude that SCHC imposes a lot less overhead than the other protocols and requires a lot less packet exchanges for a highly fragmented packet where one or more fragments were lost.

In Figure 2.9, the number of exchanged packets is illustrated for varying payloads. This figure clearly shows that packet retransmissions are much more efficient when using SCHC. If no packets are lost, 6LoWPAN may be more efficient, as SCHC in Ack-Always mode requires a separate ACK for the last fragment, on top of the CoAP ACK in the response.

2.4.1 Overhead

Figure 2.10 shows the header overhead of each protocol stack for zero retransmissions and the retransmission of 1 packet, following the diagram of figure 2.8. For the first case, the overhead quickly increases, as all fragments require a retransmission and each fragment requires the 10 byte 6LoWPAN header. CoAP block-wise transfer with 6LoWPAN compression will quickly

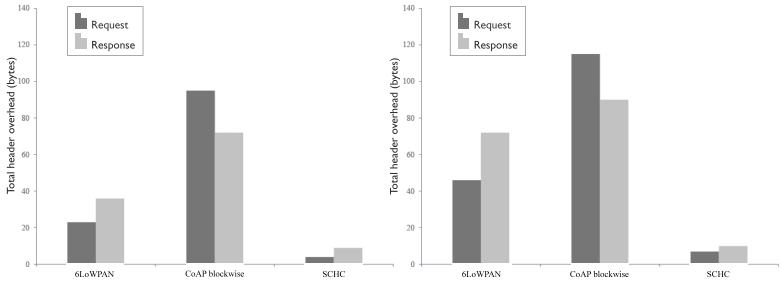


Figure 2.10: Total header overhead for a packet of length 128 and MTU 51 for different protocol configurations, left: without retransmissions, right: 1 lost fragment

perform better than the first case, as only the missing fragment is sent. However, CoAP block-wise transfer limits the size of the fragments and will therefore require 1 extra fragment in order to send the complete packet. The efficiency of the SCHC fragmentation scheme can clearly be seen from the left graph. A retransmission of 1 packet will trigger a new acknowledgment, which results in a higher response overhead, as seen in the figure on the right.

2.5 Implementation

Our second part of the evaluation evaluates our implementation of the SCHC protocol. The goal of our implementation was to have a generic library, which can be used both on top of a constrained device, as well as on a powerful server device. At the LPWAN side, the OSS-7 operating system is used [20], the application server uses the Click modular router [21] for packet processing and IPv6 forwarding. OSS-7 is designed for constrained devices, whereas Click can forward up to 333,000 64-byte packets/second [21]. The constrained nodes will mostly have one connection and will often not have a Memory Management Unit (MMU) on board. In order not to load the Central Processing Unit (CPU) with memory allocation and deallocation, preferably a fixed block of memory is used. Since Click is designed on top of a Linux kernel, memory allocation is not an issue. Apart from that, the constrained node will mostly have one connection, whereas the server will have to support simultaneous packet reassembly from multiple senders. These constraints were taken into account while designing the library, nevertheless making it as generic as possible.

2.5.1 Network Memory Buffers

Upon reception of a new fragment, the receiver will allocate a chunk of memory for that fragment. The constrained node will take a chunk of a fixed memory block, the server will allocate a chunk of memory from the heap. As none of the fragments contains the total packet length, a Network Memory Buffer (mbuf) chain is utilized, derived from the FreeBSD operating system [22], which allows trimming network headers and concatenating fragments with very little overhead. The fragment is added to the end of the linked list. Each fragment selects a slot from the mbuf pool, sets the data pointer and attaches this mbuf as the last one in the chain for this connection. In order to return a byte aligned mbuf chain to the application, with the bytes concatenated as before transmission, thus without the fragmentation headers and using the same memory buffer, the headers of each mbuf are removed and the payload of the next mbuf in the chain is shifted in.

2.5.2 Connection State

Each SCHC connection requires a certain amount of state information. The state information is kept to a minimum, in order to reduce the amount of Random Access Memory (RAM) required for each connection. Two connections are required: a TX and an RX connection. Both connections require 72 bytes of data. Connection state information is changed by the state machine, but may also be changed by the application. This can be useful when the underlying communication technology changes, which may result in a larger or smaller MTU. The library must acquire this information in order to send fragments of a correct size.

2.5.3 Timers and Retransmissions

The main loop is driven by the duty cycle timer. Once the first fragment is sent, the duty cycle timer is set, depending on the local regulations. Since the connection struct keeps track of the duty cycle time, this can be changed by the application at run-time, allowing the device to respond to dynamic requirements of the application. After an invocation of the timer, the timer calls the fragmentation state machine, in order to continue the transmission of the fragments. The implementation does not keep track of packet contents after they have been sent by the underlying LPWAN technology. Once an acknowledgment arrives, the bitmap included in the acknowledgment is used to re-generate the corresponding fragment. From an application's point of view, retransmitting data is performed the same way as the data was sent originally. Hence, the same code can be used for sending and retransmitting

data.

2.6 Evaluation

2.6.1 Connections

In order to test the implementation, we created a python script, which generates fragments for a number of devices. The fragments are sent to a Message Queuing Telemetry Transport (MQTT) broker which will forward them to the server. The experimental setup consists of a computer running Ubuntu 16.04 with 16 GB of RAM and a 2.8 GHz Intel Core i5-7440HQ CPU. We successfully evaluated the server to keep track of 100.000 device connections, including fragments and timer callback information.

2.6.2 Memory Footprint

Table 2.4 shows the memory required to implement the full standard including all reliability modes and the use of variable window sizes. Some optimization can be done, mostly for the compressor. The RAM used for the fragmenter depends upon the configuration and does not include cumulative RAM usage. The fragmenter implements the three reliability modes, padding management and the mbuf manager. The compressor implements the generic compression mechanism and a protocol parser for IPv6, UDP and CoAP. The following calculations illustrate the required memory for a constrained device. Each RX connection will add 72 bytes of RAM. Each mbuf requires 16 bytes of memory. The current `MBUF_POOL` consists of 8 mbuf's, resulting in 128 bytes of RAM. Currently, the rules are implemented in human-readable fashion, which requires 180 bytes for each IPv6 rule, 48 bytes for a UDP rule and 400 bytes for each CoAP rule. An obvious scenario would require 2 UDP rules and 2 IPv6 rules: one rule to handle the default configuration and one to handle less frequent traffic. The number of CoAP rules are very application dependent. In order to achieve a higher compression ratio, more rules are required. For this example, 7 CoAP rules are considered. An extra 3.256 bytes are thus required for this scenario. Ludovici et. al report that their 6LoWPAN implementation requires 22.584 bytes of Read Only Memory (ROM) and 3.421 bytes of RAM. [23] The compressor currently requires recursive allocated RAM which is not considered best practice, which is not included in the calculations and can sometimes cause memory overflows.

Table 2.4: Code space required for each SCHC component

	Compressor	**Fragmenter**	**Rules**	**Total**
RAM	1061 B	528 B	0 B	1589 B
ROM	6082 B	6692 B	3256 B	16030 B

2.7 Conclusion

We presented SCHC compression and its accompanying fragmentation mechanism and compared its overhead and number of packet exchanges to CoAP block-wise transfer and 6LoWPAN compression and fragmentation. Our analysis shows that SCHC is the only protocol which is able to support end-to-end IPv6 connectivity for a multimodal device, supporting SigFox, LoRaWAN and DASH-7 and therefore requires a new compression technique as proposed by the LPWAN WG. We showed that SCHC is a more suitable compression and fragmentation mechanism for LPWAN devices in terms of header overhead, reliability and total number of packets exchanged than 6LoWPAN. Our analysis can therefore be used to justify SCHC's right of existence.

Apart from that, we evaluated our SCHC implementation, which is designed in a generic fashion with a relative small footprint for an embedded device and is suited to handle a growing number of devices at the server side, nevertheless able to suit the low-memory needs of a constrained device. Compared to a 6LoWPAN implementation, SCHC also requires a lot less memory. The implementation is available as free software on https://github.com/imec-idlab/libschc.

References

[1] D. Castells-Rufas, A. Galin-Pons, and J. Carrabina. *The Regulation of Unlicensed Sub-GHz bands: Are Stronger Restrictions Required for LPWAN-based IoT Success?* page 17.

[2] U. Raza, P. Kulkarni, and M. Sooriyabandara. *Low Power Wide Area Networks: An Overview.* arXiv:1606.07360 [cs], June 2016. arXiv: 1606.07360. Available from: http://arxiv.org/abs/1606.07360.

[3] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke. *A Survey of LoRaWAN for IoT: From Technology to Application.* Sensors, 18(11):3995, November 2018. Available from: http://www.mdpi.com/1424-8220/18/11/3995, doi:10.3390/s18113995.

[4] L. Alliance. *LoRaWAN™ 1.1 Specification*, 2017. Available from: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.

[5] M. Weyn, G. Ergeerts, R. Berkvens, B. Wojciechowski, and Y. Tabakov. *DASH7 alliance protocol 1.0: Low-power, mid-range sensor and actuator communication.* In 2015 IEEE Conference on Standards for Communications and Networking (CSCN), pages 54–59, October 2015. doi:10.1109/CSCN.2015.7390420.

[6] D. Alliance. *DASH7 Alliance Specification v 1.1*, 2017.

[7] P. Ruckebusch, S. Giannoulis, I. Moerman, J. Hoebeke, and E. De Poorter. *Modelling the energy consumption for over-the-air software updates in LPWAN networks: SigFox, LoRa and IEEE 802.15.4g.* Internet of Things, 3-4:104–119, October 2018. Available from: https://linkinghub.elsevier.com/retrieve/pii/S2542660518300362, doi:10.1016/j.iot.2018.09.010.

[8] J. Finnegan and S. Brown. *A Comparative Survey of LPWA Networking.* arXiv:1802.04222, February 2018. Available from: http://arxiv.org/abs/1802.04222.

[9] L. Feltrin, G. Tsoukaneri, M. Condoluci, C. Buratti, T. Mahmoodi, M. Dohler, and R. Verdone. *Narrowband IoT: A Survey on Downlink and Uplink Perspectives.* IEEE Wireless Communications, 26(1):78–86, February 2019. Available from: https://ieeexplore.ieee.org/document/8641430/, doi:10.1109/MWC.2019.1800020.

[10] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP).* Technical Report RFC7252, RFC Editor, June 2014. Available from: https://www.rfc-editor.org/info/rfc7252, doi:10.17487/rfc7252.

[11] K. Hartke. *Observing Resources in the Constrained Application Protocol (CoAP).* Technical Report RFC7641, RFC Editor, September 2015. Available from: https://www.rfc-editor.org/info/rfc7641, doi:10.17487/RFC7641.

[12] A. Bhattacharyya, S. Bandyopadhyay, A. Pal, and T. Bose. *Constrained Application Protocol (CoAP) Option for No Server Response.* Technical Report RFC7967, RFC Editor, August 2016. Available from: https://www.rfc-editor.org/info/rfc7967, doi:10.17487/RFC7967.

[13] C. Bormann and Z. Shelby. *Block-Wise Transfers in the Constrained Application Protocol (CoAP).* Technical Report RFC7959, RFC Editor, August 2016. Available from: https://www.rfc-editor.org/info/rfc7959, doi:10.17487/RFC7959.

[14] C. Gomez, J. Paradells, C. Bormann, and J. Crowcroft. *From 6LoWPAN to 6Lo: Expanding the Universe of IPv6-Supported Technologies for the Internet of Things.* IEEE Communications Magazine, 55(12):148–155, December 2017. Available from: http://ieeexplore.ieee.org/document/8198820/, doi:10.1109/MCOM.2017.1600534.

[15] I. Ishaq, D. Carels, G. Teklemariam, J. Hoebeke, F. Abeele, E. Poorter, I. Moerman, and P. Demeester. *IETF Standardization in the Field of the Internet of Things (IoT): A Survey.* Journal of Sensor and Actuator Networks, 2(2):235–287, April 2013. Available from: http://www.mdpi.com/2224-2708/2/2/235, doi:10.3390/jsan2020235.

[16] P. Thubert and R. Cragie. *IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch.* Technical Report RFC8025, RFC Editor, November 2016. Available from: https://www.rfc-editor.org/info/rfc8025, doi:10.17487/RFC8025.

[17] P. Thubert, C. Bormann, L. Toutain, and R. Cragie. *IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header.* Technical Report RFC8138, RFC Editor, April 2017. Available from: https://www.rfc-editor.org/info/rfc8138, doi:10.17487/RFC8138.

[18] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J.-C. Zúñiga. *SCHC: Generic Framework for Static Context Header Compression and Fragmentation.* Request for Comments RFC 8724, Internet Engineering

Task Force, April 2020. Num Pages: 71. Available from: https://datatracker.ietf.org/doc/rfc8724, doi:10.17487/RFC8724.

[19] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. Request for Comments RFC4944, RFC Editor, November 2016. Available from: https://www.rfc-editor.org/info/rfc4944, doi:10.17487/RFC4944.

[20] A. University and Aloxy. *Sub-IoT: Open Source Stack for Dash7 Alliance Protocol*, September 2021. original-date: 2013-10-09T10:35:28Z. Available from: https://github.com/Sub-IoT/Sub-IoT-Stack.

[21] E. Kohler, R. Morris, B. Chen, J. Jannotie, and F. Kaashoek. *The Click Modular Router*. page 34.

[22] B. Milekic. *Network Buffer Allocation in the FreeBSD Operating System*. page 13.

[23] A. Ludovici, A. Calveras, M. Catalan, C. Gómez, and J. Paradells. *Implementation and Evaluation of the Enhanced Header Compression (IPHC) for 6LoWPAN*. In The Internet of the Future, volume 5733, pages 168–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. Available from: http://link.springer.com/10.1007/978-3-642-03700-9_18, doi:10.1007/978-3-642-03700-9_18.

3

Device Discovery and Context Registration in SCHC Networks

The previous Chapter presented the novel Static Context Header Compression (SCHC) standard and its promise to bring Internet Protocol Version 6 (IPv6) connectivity to long range networks with very high bandwidth limitations. In order to achieve this, the specification assumes that both the sensor device and the network gateway have access to the same static context. However, it is not specified how the contexts are provisioned, which limits the usability. Furthermore, in order to pave the way towards LPWA networks that can cooperate without the need to translate their Application Programming Interfaces (APIs), this chapter analyses the limitations of an offline pre-provisioning approach and presents two provisioning protocols as a solution. Both approaches provide *true* network interoperability between heterogeneous Low Power Wide Area Networks (LPWANs). Finally, the solutions are evaluated in a Light Weight Machine to Machine (LwM2M) environment, which shows the relevance of the developed solutions in a dynamic environment.

⋆⋆⋆

This chapter is based on the homonymous article by

B. Moons, E. De Poorter, and J. Hoebeke

Published in MDPI Information Journal 12(2), 16th of February 2021

Abstract Due to the limited bandwidth of LPWANs, the application layer is currently often tied straight above the link layer, limiting the evolution of sensor networks distributed over a large area. Consequently, the highly efficient SCHC standard was introduced, where devices can compress the IPv6 and upper layer protocols down to a single byte. This approach, however, assumes that every compression context is distributed before deployment, again limiting the evolution of such networks. Therefore, this paper presents two context registration mechanisms leveraging on the SCHC adaptation layer. This is done by analyzing current registration solutions in order to find limitations and optimizations with regard to very constrained networks. Both solutions and the current State of the Art (SOTA) are evaluated in a LwM2M environment. In such situation, both developed solutions decrease the energy consumption already after 25 transmissions, compared with the current SOTA. Furthermore, simulations show that Long Range Wide Area Network (LoRaWAN) devices still have a 80% chance to successfully complete the registration flow in a network with a 50% Packet Error Ratio (PER). Briefly, the work presented in this paper delivers bootstrapping tools to constrained, SCHC-enabled networks while still being able to reduce energy consumption.

3.1 Introduction

The success of the internet introduced the Transmission Control Protocol (TCP)/Internet Protocol (IP) suite as a global standard for reliable internet communication. Web page requests, file transfers, and e-mail exchanges are all built on top of this protocol suite. For applications that require less reliable, but faster, communication, such as audio and video streaming, the User Datagram Protocol (UDP)/IP pair is the perfect alternative. However, with the increasing popularity of the internet in the early 1990s, it soon became clear that the limited number of addresses the Internet Protocol Version 4 (IPv4) protocol offered would eventually run out. Network Address Translation (NAT) was created to allow the expansion of the number of internet nodes beyond the theoretical limit. However, as this was just postponing the necessity to converge to a protocol with a wider range of addresses, the Internet Engineering Task Force (IETF) started drafting the IPv6 protocol in 1998.

Apart from the almost inexhaustible range of addresses, IPv6 also aims to simplify the header format, improve support for extensions with the Internet Control Message Protocol (ICMP) for IPv6 (ICMPv6), network

configuration, and privacy [1].

The success of the internet was largely determined by the end-to-end principle, which moves the complexity of the network to the edge to allow simpler upgrades of networks and applications. The internet protocols however, were developed for high throughput Ethernet networks with multicast support at the link layer. These mechanisms are often not available in Internet of Things (IoT) networks, as sensor devices tend to be sleeping as much as possible and are often constrained in terms of bandwidth and duty cycle. Nevertheless, in order to facilitate the emergence of the IoT, where any thing is envisioned to be connected over the internet, a move towards distributed, IPv6-enabled networks is indispensable. However, for constrained devices being able to run the TCP/IP or UDP/IP suite, novel approaches are required, as the overhead brought by the current internet protocols is often too large for the available bandwidth. A first attempt in order to form IPv6 networks over IEEE 802.15.4, resulted in updated frame formats and novel network formation methods, developed by the IPv6 over Low-Power Wireless Personal Area Networks (LoWPANs) (6LoWPAN) Working Group (WG) [2].

However, the lack of bandwidth in long range communication technologies started the formation of the LPWAN WG, as there is often no room for the 6LoWPAN protocol overhead [3]. The effort of this WG resulted in the SCHC standard, which defines a generic framework for header compression.

The static nature of the proposed standard, however, requires configuration on both sides of the network, which is not in line with the autoconfiguration mechanisms provided by the IPv6 protocol. Being able to autoconfigure the smallest devices to the internet will result in easier roll-out of LPWANs and might facilitate the growth of a global LPWA network, where Commercial Off The Shelf (COTS) electronics may connect to any available LPWA network. Moreover, as the European Telecommunications Standards Institute (ETSI) recently drafted Cyber Security for Consumer Internet of Things (CYBER), endorsing security by design [4], calls for standardized connectivity, leveraging on well known protocols, which has the benefit of providing end-to-end security.

Therefore, in this paper, two solutions for IPv6 configuration and Neighbor Discovery (ND) for SCHC-enabled devices are presented and evaluated in order to provide a dynamic interface to the internet for the smallest devices available. The remainder of this paper is organized as follows. Section 3.2 gives an overview of a dynamic IPv6 configuration in long range networks. Next, the different registration mechanisms that are available today are discussed in Section 3.3. Possible solutions build upon this and are presented in Sections 3.4 and 3.5. Finally, both solutions are evaluated in Section 3.6

and the advantages and disadvantages for both solutions are discussed in Section 3.7.

3.1.1 Related Work

Static Context Header Compression started receiving increasing attention from the research community. In both [3] and [5], the compression mechanism is implemented and evaluated, in C and NS-3, respectively. The authors of [3] also implemented the fragmentation mechanism and made the library publicly available [6]. Both authors conclude that a novel adaptation standard is needed, as 6LoWPAN does not provide enough flexibility for LPWAN communication technologies. The authors of [5] continue their work in [7], where a central Administration Management Server (AMS) manages the context in order to support roaming of devices between different LoRaWAN operators. While this approach allows devices to roam between multiple networks of different operators, the AMS, and consequently complex architectural components, are still required to keep track of the device rules. Bernard et al. [8] argue that the AMS scenario is only possible when operators have a link between each other to retrieve the location of the AMS and propose the use of Domain Name System (DNS) in order to download the rules from an Hyper Text Transfer Protocol (HTTP) server. This way, the rules can be managed remotely and do not require storage at the Network Gateway side. However, no solution is given to synchronize the updated rule context between end devices and the Network Gateway, and therefore there does not exist a solution to dynamically install the context at the end-device.

The limitations of a context that was distributed before deployment is something which was also quoted in [9]. The authors proposed a novel matching operator, i.e., the dummy-mapping Matching Operator (MO), which allows the target value in the rule to remain unknown for the sensor device and editable on the network gateway side. Values that are not important for the device, but that are required to match requests and responses, could make use of this matching operator, e.g., IPv6 source address and UDP source port. However, in order to completely manage such networks, a more advanced solution is required. Furthermore, to manage bidirectional IPv6 communication, the SCHC gateway should be aware of the connected devices, which is currently not defined in the standard.

3.2 Motivation

Due to their ability to offer low-power connectivity for devices distributed over a large area, Low-Power Wide-Area Networks are increasingly gaining attention. Engineering companies are in the attempt of developing a communication technology to serve as many use cases as possible. Consequently, many communication technologies are currently available, such as IEEE 802.15.4, SigFox, LoRaWAN, Weightless, NB-Fi, Narrowband IoT (NB-IoT), enhanced Machine Type Communication (eMTC), Extended Coverage Global System for Mobile Communication (GSM) (EC-GSM), and DASH7, among others [10]. This plethora of things and technologies will enable, for example, ports, logistics, cities, and agriculture to become smart. Therefore, a multitude of heterogeneous devices and technologies will have to be integrated in the overarching Information System using proprietary APIs. Maintenance to catch changes to the APIs and to integrate them in a single back-end platform becomes a burden to the application developer and leads to scalability issues. This also happened in early Wireless Sensor Networks (WSNs), where the application layer was tied straight above the data link layer [11]. However, due to the efforts of the 6LoWPAN WG, middle-boxes of service providers could be decoupled, which led to easier deployment of applications and emergence of new businesses.

Figure 3.1 illustrates two LPWAN technologies that make use of a star topology, in which one or more gateways are forwarding Layer 2 packets to a central entity. The delivery of the data in a proprietary way results in vertical silos, which is in sheer contrast with the open vision of the internet, where a device can address any other host directly. By deploying SCHC at the gateway (or network server), these architectures can evolve to more scalable networks, as depicted in Figure 3.2. This, however requires a bootstrapping protocol in order to perform context and device discovery.

Secondly, promising uses for LPWAN technologies can include remote monitoring of assets in the logistics industry in order to build a resilient supply chain. As logistics are characterized by international shipments, infrastructure will most probably be delivered by multiple LPWAN providers. Recent advancements in the LoRaWAN landscape, for example, launched a secure pre-provisioning platform, which offers a network agnostic Join Server. This approach allows for secure end-devices to become decoupled from the network they are using [12]. This way, LoRaWAN devices could securely roam between different LoRaWAN networks. In the architecture of Figure 3.2, such gateways will manage an IPv6 subnet in which sensor devices can obtain a unique global IPv6 address. This, however, requires every component in the network to be aware of the SCHC configuration. The

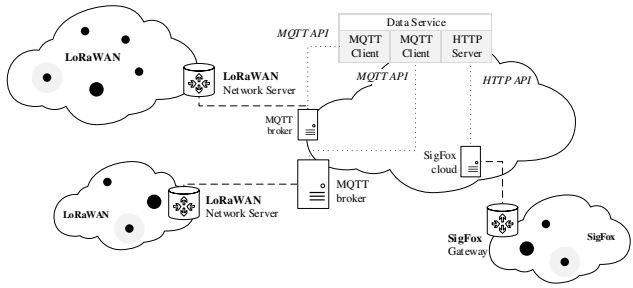


Figure 3.1: The current Low-Power Wide-Area Network landscape: centralized components deliver data to translation units.

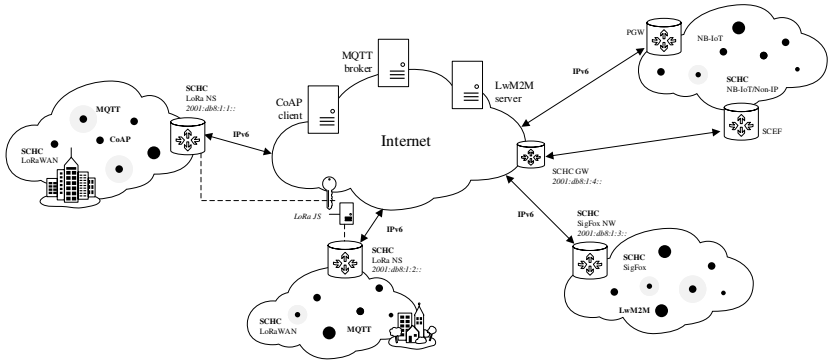


Figure 3.2: The envisioned LPWAN landscape: end-to-end secured, decentralized IPv6 enabled networks.

current SCHC standard does not allow such dynamic configuration, which would result in a per-packet header overhead, due to (partially) uncompressed headers.

Thirdly, despite the fact that the task of these sensor devices can often be reduced to the sensing and reporting of their environment, these rather static configurations can benefit from dynamic context configuration. LwM2M devices, for example, can be pre-provisioned with the IPv6 address of the Bootstrap Server. Only after a bootstrap request to the LwM2M bootstrap server, the endpoint will be provisioned with the LwM2M Server object(s).

This reply contains their IPv6 address(es), requiring configuration of the lower layers and consequently the SCHC context. Furthermore, the address of any given end-point may change, which requires dynamic configuration of devices and intermediaries, and their context.

In order to solve the above issues, SCHC devices should be able to register themselves and their compression context to the corresponding Network Gateway, which is currently not included in the standard. Therefore, the current SOTA regarding device registration and management in IPv6-enabled networks is analysed in the next section in order to determine possible gaps of these protocols in SCHC enabled networks.

3.3 Device Management and Registration

Today, several solutions exist to manage devices on a network. In traditional IPv6 networks, the ICMPv6 protocol is the main source of control. Apart from error messages and informational messages, several extensions exist in order to control such networks. The ND, for example, allows devices to discover routers on the network and register their link-layer address and IPv6 address so routers know where to forward their packets to. Extensions are used in LoWPANs to optimize these protocols for networks that have a limited amount of bandwidth available and are often restricted by a regulatory duty cycle. For other purposes, several network management protocols exist, such as the Network Configuration Protocol (NETCONF) protocol, which can be used to monitor networks and manage devices and their interfaces.

3.3.1 Basic Neighbor Discovery Protocol

In IPv4, the Address Resolution Protocol (ARP) and ICMP Router Discovery and Redirect were introduced in order to register a node to a Local Area Network (LAN). The IPv6 WG reused parts of these functionalities and combined them with new mechanisms to detect neighbor unavailability and the presence of duplicate IP addresses to form the ND specification [13]. These standardization efforts are used by IPv6 nodes for

- neighbor detection: determination of the layer 2 address of nodes on the same link,
- router discovery: discovering neighboring routers that can forward their packets, and
- neighbor unreachability detection: actively keeping track of changing neighbors.

In traditional IPv6 ND, a wireless device will transmit a Router Solicitation (RS) as a multicast to request the network prefix from the network it moved to. The routers on the network will reply to the all nodes multicast address with a Router Advertisement (RA), containing several ICMPv6 options, such as the Maximum Transfer Unit (MTU) of the link, the default route and the network prefix for subnet information. The device will then autoconfigure its link-local and global address(es) if the Managed address configuration (M) flag is set. It will perform Duplicate Address Detection (DAD) for each address it assigned itself, by multi-casting a Network Server (NS) including the DAD option for each address. Once a node wants to reach a different node on the network, it will also send out a NS message to determine its link-layer address or to verify its reachability state. This is called Neighbor Unreachability Detection (NUD) and is answered with a Neighbor Advertisement (NA) message.

This set of mechanisms could be reused for address registration in LPWANs. However, the overhead and the heavy use of multicast in IPv6 networks is impractical for LPWANs that are composed out of constrained and sleepy end-devices, connected over a constrained wireless link. Previous evolutions in WSNs had to cope with similar issues and adapted the traditional model to the needs of low-power networks. Consequently interesting features were added in the 6LoWPAN standard [14].

3.3.2 Optimized Neighbor Discovery Protocol

The basic IPv6 Neighbor Discovery (ND) mechanism has been optimized for 6LoWPAN networks, which resulted in Request For Comments (RFC) 6775: Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks, which is, however, not limited to LoWPANs. A more generic approach was presented in [15], where Neighbor Discovery was applied to MultiLink IoT subnets and a backbone was incorporated to perform neighbor discovery on behalf of these low-power devices from multiple mesh networks. This optimization mainly consists of eliminating NS multicast messages to all other nodes in the network and periodic RA messages from routers. The RS/RA message pair in optimized ND is now used to request prefix and context information from the router. Every RS message creates or updates a Neighbor Cache Entry (NCE) with a certain lifetime, which should be updated by the hosts themselves with a RS message. On the other hand, the NS/NA message pair is used to perform address registration, DAD and NUD by a central IPv6 ND Registrar on behalf of the host [16]. The Neighbor Registration and Periodic Updates section of Figure 3.3 give an overview of the optimized ND protocol.

Optimized ND inherits all functions defined in the ND specification,

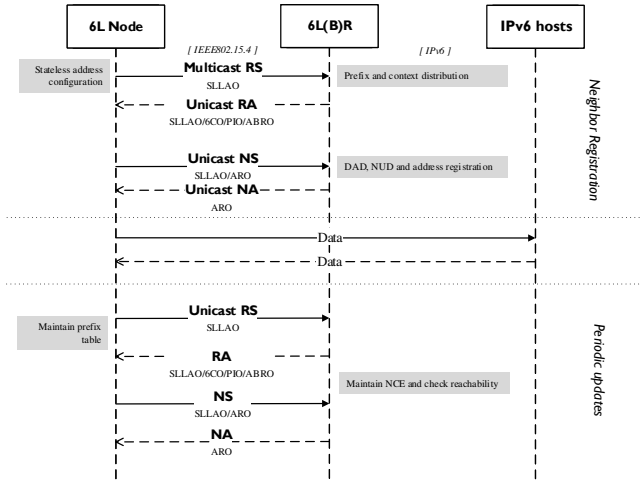


Figure 3.3: 6LoWPAN optimized neighbor discovery. Nodes regularly cast re-registration requests in order to keep their NCE fresh.

however, introduced several optimizations in order to cope with the low power nature of LoWPANs.

3.3.2.1 Address Registration Option

The registration specified in 6LoWPAN-ND is a unicast message between the device and the 6LoWPAN router with the use of the new Address Registration Option (ARO). Upon reception of a RS message, the router may create a tentative NCE with a specific timeout for the requested Source Link-Layer Address Option (SLLAO). It will respond with an NA message indicating if a duplicate address was detected or if the address registration process was successful.

3.3.2.2 Prefix and Context Information Distribution

A 6LoWPAN host joining a network will send out a Router Solicitation message to solicit information about the network from active routers on the network. The host will extract the received context and subnet information from the border router, which can later be used for the decompression process.

3.3.2.3 Others

Network Parameter Discovery is not used in Optimized ND, as network information is distributed in response to a RS message sent by the host while requesting prefix and context information about the network. Furthermore, DAD and Address Resolution is not performed separately, as this is part of the address registration process. Finally, in order to keep track of changes in the network, hosts periodically multicast NS messages to confirm other nodes their reachability state. In optimized ND however, hosts are prevented from sending multicast NS messages, this information is now taken care of by the ARO in the NS messages they send. Before the NCE expires, a RS message is sent to detect if one of the default routers have become unavailable.

3.3.3 NETCONF

Apart from the ICMPv6 protocol to distribute information in IPv6 networks, other management protocols to manage routers, switches, and modems came into being. In 1988, the Simple Network Management Protocol (SNMP) was standardized as a protocol for managing IP networks. Due to its relatively simple architecture, it could be deployed on different kinds of networks and became one of the most widespread network management protocols [17]. However, as SNMP was not developed with programmability in mind, the IETF started the development of NETCONF in order to ease configuration of devices on a network. NETCONF makes use of Remote Procedure Call (RPC) encoded in eXtensible Markup Language (XML) to retrieve or edit a configuration datastore. The data are validated using Yet Another Next Generation (YANG). YANG defines a data model, which formats like XML must adhere to. NETCONF then defines a set of operations (Create, Read, Update, Delete (CRUD)), used to access and update the datastores of devices connected to the network. As NETCONF runs over a Secure Shell (SSH), the IETF developed RESTCONF to access datastores and manage network equipment over HTTP by means of web applications. The HTTP POST, PUT, GET, and DELETE methods can be mapped easily to NETCONF's CRUD operations.

3.3.3.1 CORECONF

Similarly, the IETF started the development of the CoAP Management Interface (CORECONF), mapping the various CRUD operations of NETCONF to the methods available in the Constrained Application Protocol (CoAP) to manage constrained devices [18]. In order to limit the protocol overhead, YANG structured messages are encoded using the Concise Binary Object

Representation (CBOR). Recently, a YANG data model for the SCHC compression and fragmentation rules has been submitted as an Internet-Draft. The goal of this document consists of formalizing the description of the rules to offer interoperability among implementations and a manner to update specific values on either end [19].

3.3.4 LwM2M

Another IoT management protocol has been developed by the Open Mobile Alliance (OMA) and goes under the name of Light Weight Machine to Machine (LwM2M) [20]. The LwM2M protocol defines objects, resources, and instances on top of CoAP in order to limit the protocol overhead when querying well-known resources from a sensor device. Apart from information reporting, also bootstrapping, device registration and ways to perform device management are present in the specification.

3.3.5 Conclusion

As outlined in this Section, several solutions exist today to manage low-power networks in IPv6 environments. However, current solutions available in ICMPv6 for configuration of a context are insufficient to realize what SCHC is trying to achieve; optimal efficiency in terms of protocol overhead. The IETF has therefore set up initiatives in order to manage the context of SCHC enabled devices. CORECONF, however, forces SCHC devices to implement CoAP as an application layer protocol to manage the lower layer protocol(s), whereas SCHC, as a generic framework, targets any kind of protocol. Therefore, this paper presents two alternative solutions that can be used with any protocol; one purely built on SCHC and the other based on the ICMPv6 protocol.

3.4 Device Registration

The first problem that arises when connecting LPWAN devices to the internet using the SCHC protocol, is the inability of such devices to dynamically create an IPv6 address. Consequently, the SCHC router is not able to capture downlink packets destined for the sensor node until an entry is created in its Neighbor Discovery Cache. Routers need to know the host IP addresses of the sensor nodes and their corresponding link-layer addresses. This also has to be maintained as their reachability might change. Regular IPv6 routers use the NS/NA message pair to map link-layer addresses to IPv6 addresses. 6LoWPAN routers maintain a NCE for the duration of a lifetime included in the ARO. Others, such as Source Address Validation

Improvement (SAVI) tables, build a binding table to proxy ND on behalf of the device in order to dampen multicast usage in wireless networks [15]. SCHC could also benefit from a mechanism where the SCHC router takes care of common IPv6 actions on behalf of the sensor nodes. Therefore, two solutions to keep track of device registrations and queued downlink traffic are proposed in the next Sections. The first one, called SCHC Registration, is an extension to the SCHC protocol. The second one is an entirely standards based solution that makes use of the ICMPv6 Neighbor Discovery protocol. Both mechanisms make use of a new SCHC component in order to limit the protocol overhead: the SCHC Rule Registry (SRR).

3.4.1 SCHC Rule Registry

The SRR is a well-known repository, which defines the most common actions to perform different management operations. All SRR compliant components must be pre-provisioned with this standardized context. Consequently, these SCHC identifier values are forbidden for application specific rules. Depending on the employed solution, the implementation of the context differs and will be further explained in the next sections. Every compression action in the SRR consists of four entries, as each reliability mode requires a separate rule id. However, as some implementations might prefer a static solution, not too many SCHC ids should be reserved. Furthermore, RFC 8724 suggests a variable sized identifier and lets an application or technology fix it. Very constrained technologies, such as Sigfox, recommend a 3- or 4-bit rule id for regular packets [21]. Therefore, higher order identifier values, i.e., ids 32 up to 63, are reserved for the SRR. This way, devices with limited space for extra overhead can still make use of the smallest rule ids possible and further extend their id range to support device registration. As a consequence, this extension requires a minimum overhead of 6-bit rule id.

3.4.2 SCHC Registration

The first solution consists of an extension to the SCHC protocol, which we called SCHC Registration. In order for the already standardized SCHC components (i.e. the compressor/decompressor and the fragmenter/reassembler) to work independently from the registration mechanism, a third abstraction layer is added to the architecture: the SCHC Context Manager (SCM). The SCM takes care of device and context registrations and does so by inspecting every incoming SCHC packet in order to determine the action to perform, based on the employed rule id. These actions are stored in the SRR and are defined as follows.

Table 3.1: SCHC registration request.

Field	FL	DI	TV	MO	CDA
...	...	...	...	...	...
IPv6 Next Header	8	BI	17	`equal`	`not-sent`
IPv6 Src Prefix	64	UP	::	`equal`	`not-sent`
IPv6 Src IID	64	UP	::	`equal`	`not-sent`
IPv6 Src Prefix	64	DO	::	`ignore`	`value-sent`
IPv6 Src IID	64	DO	::	`ignore`	`value-sent`
IPv6 Dst Prefix	64	UP	::	`ignore`	`value-sent`
IPv6 Dst IID	64	UP	::	`ignore`	`value-sent`
UDP Src Port	16	BI	-	`ignore`	`value-sent`
UDP Dst Port	16	BI	-	`ignore`	`value-sent`
UDP Length	16	BI	-	`ignore`	`compute-*`
UDP Checksum	16	BI	-	`ignore`	`compute-*`

1. Registration: identifiers (32–35) used to register a device and the IPv6/UDP endpoint it wishes to communicate with.

2. Extended registration: identifiers (36–39) can be used to provide more flexibility during registration.

3. Re-registration: identifier (40) used to keep the device and SCHC gateway synchronized.

4. Bindings: set of identifiers (48–51) used as an updatable IPv6/UDP pair for context registration.

3.4.2.1 Registration

A sensor device can send a registration message by setting the source IPv6 address to an unspecified one. As Table 3.1 indicates, the reserved registration rule from the SRR will be employed by the compressor and the values of the IPv6 Destination, UDP Source Port, and UDP Destination Port are sent to the receiving gateway. This information can be used on the other side to configure the context, as will be explained in Section 3.5.

As depicted in Figure 3.4, every SCHC packet arriving at the Network Gateway, is decompressed first. The return value of the decompressor is then passed to the SCM, where either

1. a re-registration is sent if desynchronization occurred;

2. the values sent by the sensor device are saved in a context, an IPv6 address is generated on behalf of the client and distributed back if the device was not registered yet; or

3. the decompressed packet is forwarded to the IPv6 internet if the device was registered.

If a device was not registered yet, an IPv6 address, based on the device's EUI-64 address, is generated and binding rules are created. Binding rules are reserved rules that keep track of the IPv6/UDP context. Furthermore, a NCE is added for the device in order to answer ND messages on behalf of the device, queue downward packets and keep track of the registration process. In order to inform the sensor device about the network's prefix and given IPv6 address, the message is replied to using the same rule id.

Figure 3.4 shows the Finite State Machine (FSM) for traffic flowing in the downward direction. If a device, such as a Sigfox or LoRaWAN Class A device, only has uplink-triggered downlink opportunities, a state indicates if the device is reachable or not. Once a packet arrives from the IPv6 network, this state determines if a packet is queued or immediately forwarded. The device state is updated every time a packet from the sensor device arrives and is removed if no packet was received in time. The registration time is technology or application specific. This way, the SCHC gateway will notice disconnection from the device. If no device state is present on the server, a (re-)registration request is sufficient to cope with desynchronization.

If an error occurred during registration, no prefix information will be delivered in downlink, as the packet got discarded or went missing. Erroneous transmissions in downlink will force end devices to remain in the `unregistered` state. This way, the SCM will keep transmitting registration packets until it has successfully registered to the network. A safety mechanism can be used in order to limit the amount of registration messages.

3.4.3 SCHC Optimized Neighbor Discovery

The other proposed solution builds entirely on existing standards and uses SCHC to compress Optimized Neighbor Discovery messages. In order to perform ND, the SCHC router implements a Neighbor Cache—similar to

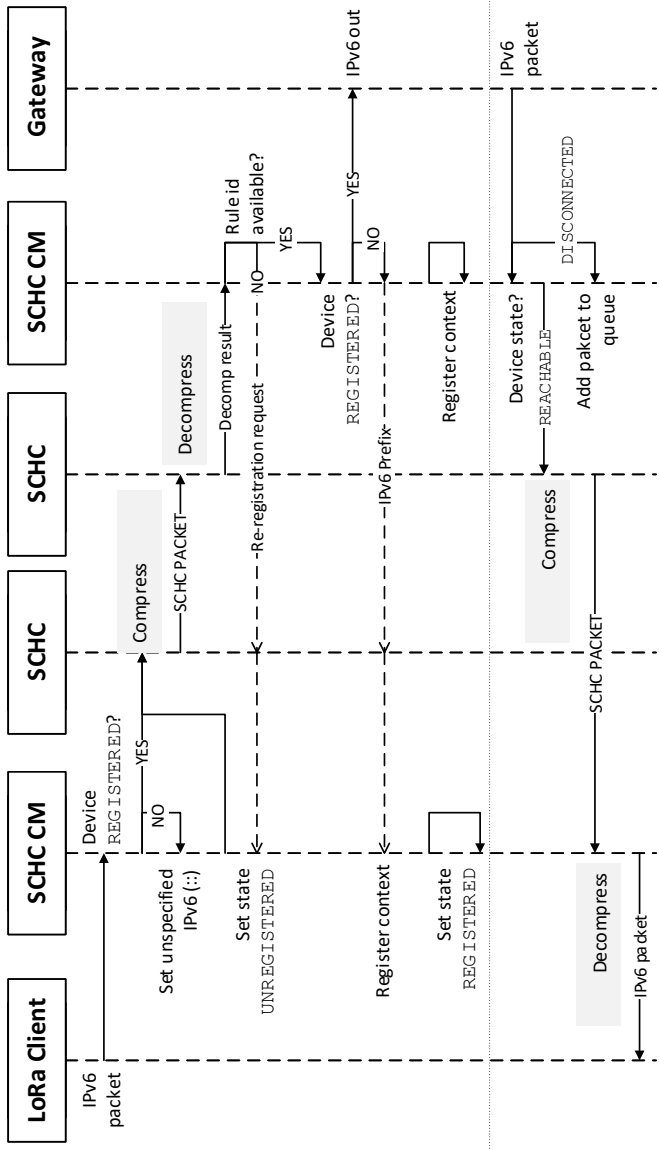


Figure 3.4: A time diagram that shows incoming and outgoing IPv6 packets in both directions for the proposed SCHC Registration mechanism. Note how the Context Manager is used as a separate layer.

6LoWPAN routers—which behaves as a registry for all host addresses attached to the router. In order to request IPv6 prefix information of the network, a sensor device configures a link-local IPv6 address and broadcasts a RS. The RA response from the gateway contains a SLLAO, MTU, and Prefix Information Option (PIO). The PIO can be used to perform stateless address autoconfiguration. Next, a NS carrying the ARO option is sent to the SCHC gateway in order to register an IPv6 address and a lifetime. The lifetime should be chosen reflecting the behavior of the node, i.e., the registration should be maintained even while the host is sleeping. This method therefore assumes that sensor devices repeat the ARO before their lifetime runs out. The complete flow for performing Neighbor Discovery and updating the SCHC context (which will be explained in the next Section) is given in Figure 3.5. Once a sensor device starts the Neighbor Discovery process by sending a RS, its state machine behaves such as regular Optimized ND. A timeout lets the device notice a lost or erroneous Router/Neighbor Solicitation or Router/Neighbor Advertisement. The responsibility for a retransmission strategy lies on the device.

In order to limit the protocol overhead of the Neighbor Discovery, the following actions should be incorporated in the SRR:

1. Router Solicitation/Router Advertisement: identifiers (32–35) used to probe for the network's prefix and parameter information.

2. Neighbor Solicitation/Neighbor Advertisement: identifiers (36–39) used to maintain an entry in the router's Neighbor Cache.

3. SCHC Control: identifiers (40–43) used to configure the context of a device.

4. Default IPv6/UDP rule: identifiers (44–47) used to minimize the context configuration overhead.

A detailed explanation of the SCHC-Optimized Neighbor Discovery requirements is given in Appendix A.1.

3.5 Context Configuration

The second problem that arises during the lifetime of SCHC devices, is the inability to configure their context. Therefore, both solutions incorporate a mechanism to configure the device context.

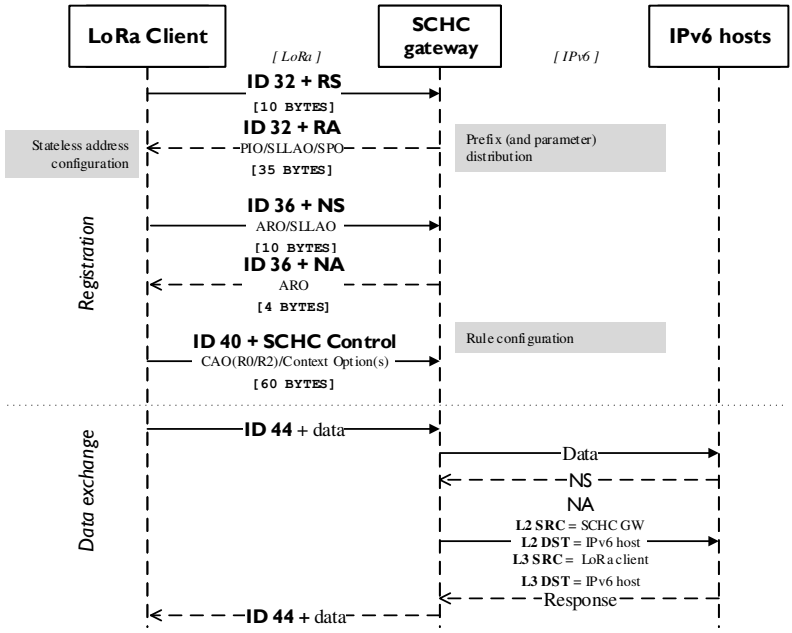


Figure 3.5: The proposed ICMPv6-based SCHC device and context registration.

3.5.1 SCHC Registration

Using SCHC Registration, a device sends a registration message to create a NCE at the corresponding SCHC router. This message also carries the values of the IPv6 Destination, UDP Source Port and UDP Destination Port. These values are used on the receiving side to update the binding rules of that particular device. Binding rules are part of the SRR and can be updated every time the device employs the registration rules. Once a device has registered its context, it can use the binding rules to send compressed packets to the SCHC router.

3.5.1.1 Extended Registration

In order to have more control over the configuration of the registration, a set of identifiers is dedicated to inform the SCHC gateway about every IPv6 and UDP field. This has a higher overhead, but improves flexibility. The Compression Decompression Action (CDA) for every IPv6 and UDP header value, except for the Version and upward Source IPv6 fields, is set to `value-sent`. This way, all header values will be communicated once and remain completely compressed afterwards.

3.5.1.2 Fragmentation

As the link towards LPWAN devices is often very constrained in terms of bandwidth, the SCHC standard also defines a fragmentation mechanism to cope with large packets. The fragmentation and reassembly procedure relies on a range of parameters, known a priori to both sides of the network. This method assumes a static configuration of these parameters, available in the profile of the underlying technology. Future work might include a proper negotiation mechanism, where the first message of the registration procedure could initialize the different fragmentation parameters.

3.5.1.3 Application Layer Compression

In order to compress application layer messages, we propose to perform double compression. Therefore, the IPv6/UDP layer and the application layer are compressed independently from each other. The SCHC gateway performs outer (de-)compression, while the receiving SCHC application performs inner (de-)compression. This way the application layer protocol can remain implementation and use case specific.

A visualization of a LoRaWAN Class A device performing the different steps in the SCHC registration process is given in Figure 3.6.

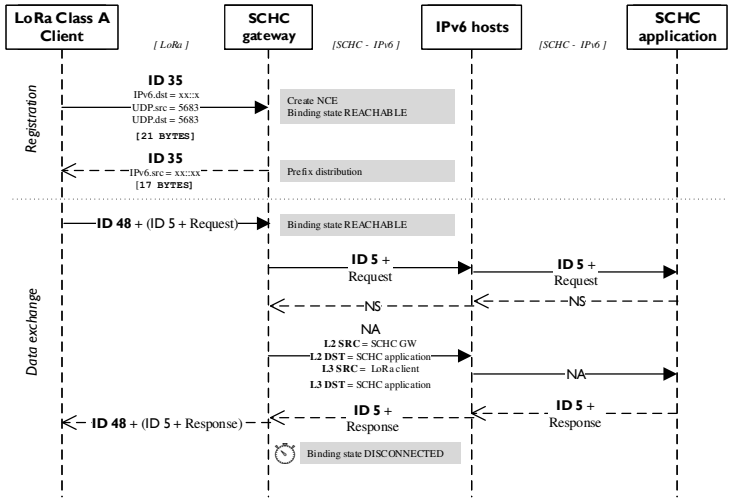


Figure 3.6: SCHC-based device and context registration. The SCHC gateway keeps track of the sensor IPv6 addresses that are connected to the gateway. On a local network, it will respond to NS requests with its own Medium Access Control (MAC) address.

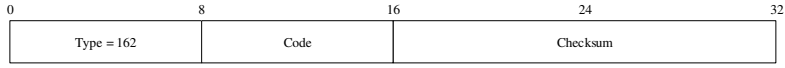


Figure 3.7: The proposed ICMPv6 header for SCHC control messages.

3.5.1.4 Limitations

The proposed SCHC registration mechanism currently forces the device to use IPv6 in combination with UDP. As RFC 8724 was baptized *a Generic Framework for Static Context Header Compression and Fragmentation*, this is not completely in line with its vision, i.e., the registration mechanism cannot be expanded towards the TCP. Furthermore, the specification of a Matching Operator or a CDA for a Target Field is not possible. Therefore, the `ignore` MO is always used in combination with `not-sent` CDA. This, however, achieves the highest compression rate. Furthermore, the proposed solution does not provide a proper way of configuring the application layer protocol and forces the device to perform double compression and the end-point to implement an SCHC layer between the transport- and application layer.

3.5.2 SCHC Control Messages

A second way of dealing with SCHC context configuration is by means of ICMPv6 control messages. The SCHC Control Message is proposed as a new ICMPv6 message in order to be able to configure rules in both directions. Figure 3.7 illustrates the structure of the SCHC Control Message.

The code field from the ICMPv6 header can be used to indicate different modes of operation:

- 0x00 indicates the creation of a new SCHC rule
- 0x01 indicates an update request of one or more fields from an existing rule
- 0x02 indicates the removal of one or more fields from an existing rule

3.5.2.1 Context Advertisement Object and Context Option

In order to indicate which rule is targeted, the Context Advertisement Object (CAO), illustrated in Figure 3.8, can be attached to a SCHC Control Message.

Currently, rule ids up to 12 bits, or a total of 4.096 rules, are supported. The length field is used to indicate the number of Context Options, which

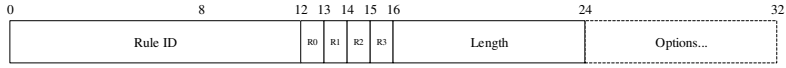


Figure 3.8: The proposed SCHC context advertisement object.

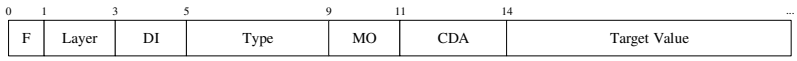


Figure 3.9: The proposed SCHC fixed size context option.

can be up to 256. This is required to know when Fragmentation Context Options will follow, which will be introduced in Section 3.5.2.3. Furthermore, the following flags are present to indicate for which reliability mode(s) a rule must be created:

1. `R0`: No Fragmentation
2. `R1`: No-Ack
3. `R2`: Ack-on-Error
4. `R3`: Ack-Always

In order to remain efficient while assigning rule ids, the rule id is increased by 1 for every R-flag. An example is given in Listing 3.1.

Listing 3.1: Usage of reliability mode flags

```
Rule ID = 25; R0 = 1; R1 = 0; R2 = 1; R3 = 1;

Compression Rule ID = 25;
Ack-on-Error Rule ID = 26;
Ack-Always Rule ID = 27;
```

CAOs that will create or update must be followed by one or more SCHC Context Options, used to represent a rule entry. In general, headers can be either variable or fixed sized. CoAP, for example, can use variable sized option fields of which the size is sent with the Compression Residue [22]. Therefore, the SCHC Context Options are divided in Fixed Size Context Options and Variable Size Context Options, providing more flexibility and efficiency in terms of header overhead. Their structure is given in Figures 3.9 and 3.10.

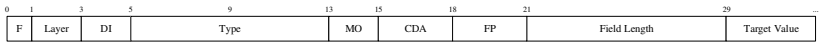


Figure 3.10: The proposed SCHC variable size context option.

Both message structures have a Fixed flag field, indicating either the fixed (0) or variable (1) type. The Layer field indicates which layer is targeted; (0) the network layer, (1) transport layer, or (2) application layer. This generic approach leaves room for inclusion of other protocols in the future. Next, every option contains the different SCHC rule columns, such as Direction, MO, CDA, and Target Value (TV). The Variable Size Context Option uses the Field Position (FP) and Field Length in order to represent multiple fields of the same type with a variable length. Finally, a specific row (field) in a SCHC rule can be targeted using the Type field. A more in depth overview of the structure is given in Appendix A.2.

In order to cope with non-byte aligned set of options, padding bits are added at the end. Furthermore, to limit the total overhead, the ICMPv6 Control Message can be compressed using rules from the rule registry.

3.5.2.2 Synchronization

Once a device resets, it will automatically re-register itself using Neighbor Discovery and reconfigure its context. In case of a router reset however, the router will reply with an ICMPv6 Type 1 error: Destination Unreachable. The code field must be set to 0x07, indicating an Error in Source Routing Header. This way, devices are informed that their contexts are desynchronized. Furthermore, once a node is de-registered from the network, i.e., the lifetime expired, its context is removed.

3.5.2.3 SCHC Parameter Option

So far, only the configuration of compression parameters were discussed. In order to configure the fragmentation layer of either the device or the network gateway, two ICMPv6 extensions are proposed in this section. Currently, devices are grouped in profiles based on a technology or a product in order to configure similar fragmentation parameters. However, to overcome the limitations imposed by this static approach, we introduce a new IPv6 Neighbor Discovery Option Format. The SCHC Parameter Option (SPO) (type 41) can be included in a Router Advertisement if the network supports the SCHC fragmentation mechanism. The structure of the SPO is given in Figure 3.11. The different fields carry several parameters that must be defined in a SCHC profile. A more detailed explanation of the different fields

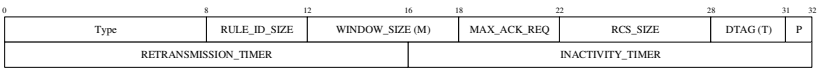


Figure 3.11: The proposed ICMPv6 SCHC parameter option.

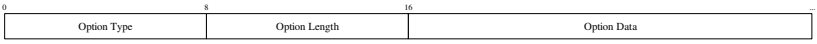


Figure 3.12: The proposed ICMPv6 SCHC fragmentation context option.

is given in Appendix A.3.

Not all parameters can be configured using the SPO. Therefore, several parameters use default values. The `MAX_PACKET_SIZE`, for example, defaults to 1500. The `FCN (N)` value defaults to 1 for the No-Ack reliability mode, and for other reliability modes it is set to the number of bits required to represent the `WINDOW_SIZE`. However, if a device desires to change the fragmentation parameters of a rule individually, the SCHC Fragmentation Context Option can be used together with a CAO and use Type-Length Value (TLV) encoding, as shown in Figure 3.12.

These options affect every targeted reliability mode by the CAO and contain the following fields:

- Option Type: 8-bit identifier of the type of option.
- Option Length: 8-bit unsigned integer representing the length of the Option Data field.
- Option Data: a variable length field that contains data specific to the option.

These types are, however, out of the scope of this paper and may be looked at in future work.

3.6 Evaluation

In this Section, a set of experiments and simulations are given in order to evaluate the performance of the protocol and to demonstrate how a real-world situation can benefit from the developed solution. All experiments where implemented in Matlab, using a rule id of 6 bits and a window size of 7 with a Fragment Count Number (FCN) of 3 bits. No Datagram Tag (DTag) was used and a Reassembly Check Sequence (RCS) of 32 bits was used during fragmentation sequences.

3.6.1 Comparison

To start with, a comparison of both solutions is given in Table 3.2. The overview clearly indicates the flexibility of the SCHC ICMPv6 Control Messages, where up to 4.096 rules can be configured. Every rule can contain at least 2.048 entries (256 types, 8 field positions per type), which can be added, updated, and removed. The proposed SCHC Registration mechanism, on the contrary, only has 16 editable fields for a single IPv6/UDP rule. Furthermore, the SCHC Registration requires 16 reserved rule id's, while this is optional for the ICMPv6 Control Messages. Furthermore, every field present in a rule can be added, updated, or deleted using the ICMPv6 method, while the SCHC Registration mechanism can only update the Target Value of the IPv6/UDP binding rule. Furthermore, the extensibility of the ICMPv6 Control Messages to any protocol makes it future-proof and robust to changes and evolution of networks. This control, however, comes with a larger transmission and energy overhead, which will be shown in the next Section.

3.6.2 Registration Overhead

In order to show the difference between the different registration mechanisms, we calculated the total overhead for uplink and downlink communication, illustrated in Table 3.3. The impact of the fragmentation overhead is also illustrated for devices (e.g., Sigfox with an MTU of 12 bytes) using the `ack-on-error` reliability mode. For the SCHC Registration mechanisms (Registration and Extended Registration), the total overhead ranges from 17 bytes up to 34 bytes. This is mainly due to the fact that the IPv6 source and destination are sent to the receiving gateway. The `ack-on-error` mode increases the overhead mainly due to the presence of the RCS, which was set to 32 bits. Both mechanisms, however, still fit the MTU of an Long Range (LoRa) Spreading Factor (SF) 12 device, without the need for fragmentation.

Depending on the request/response sequence and possible fragmentation, a higher number of packets is exchanged. This is illustrated in Figure 3.13. SCHC Registration only requires two packets (1 up and 1 down) for devices with a higher order bandwidth, such as LoRaWAN and DASH7. Sigfox devices, on the contrary, require 7 packets (3 up, 4 down). The other solution, SCHC compressed Neighbor Discovery, can be seen to be very efficient in terms of uplink communication; only 2 packets, 1 for the RA and 1 for the NA, are required for any type of device. However, 7 packets are required in downlink over a Sigfox Ultra Narrow-Band (UNB) link. This, however, could be reduced to 5 if the fragmentation parameters

Table 3.2: Comparison of the SCHC Registration and ICMPv6 SCHC Control mechanisms.

	SCHC Registration	**SCHC ICMPv6 Control Messages**
IPv6 standards compliant	No	Yes
Prefix dissemination	Yes	Yes
Compression rule registry	Mandatory	Optional
Fragmentation rule registry	Mandatory	Optional
Fragmentation parameter distribution	No	Yes
Configurable rules	1	4.096
Configurable entries per rule	16	2.048
Supported editable fields	Target value	Rule Id, Direction, Field Length, Target Value, MO, CDA, Field Position
Network layer support	Yes, up to 1 rule	Yes, up to 4.096 rules
Transport layer support	Yes, up to 1 UDP rule, not expandable without modification	Yes, up to 4.096 rules, expandable to TCP without modification
Application layer support	No, perform double compression	Yes, expandable to any protocol

are distributed a priori, and therefore no SPO must be distributed. The packet overhead for regular Optimized Neighbor Discovery is also given for reference. It can be seen that using the proposed compression mechanism, this becomes a lot more efficient.

3.6.2.1 Registration Time

As transmission errors and the impact of the regulatory limitations on duty cycle limited technologies will affect the time required to perform a registration attempt, we present a simulation model based on the PER. The PER of a network depends, among others, on the amount of gateways, the interference of other technologies, the number of devices in a network, and the back-off time between transmissions [23]. Highly dense networks have

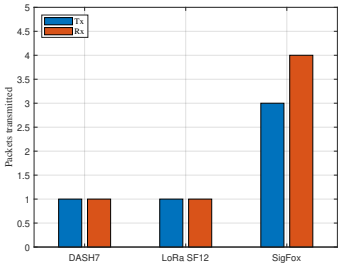
Table 3.3: Overhead in bits for the proposed specifications. The Non-fragmented mode is used for a LoRa SF12 device, while the Ack-on-Error mode is used for a Sigfox device.

	Mode	Direction	Overhead (B)
Registration	Non-fragmented	Up	21
		Down	17
	Ack-on-Error	Up	28
		Down	25
Extended R.	Non-fragmented	Up	26
		Down	17
	Ack-on-Error	Up	34
		Down	25
SCHC RA/RS with PIO/S-LLAO/SPO	Non-fragmented	RS	10
		RA	35
	Ack-on-Error	RS	14
		RA	44
SCHC NA/NS with ARO/SLLAO	Non-fragmented	NS	10
		NA	8
	Ack-on-Error	NS	10
		NA	5

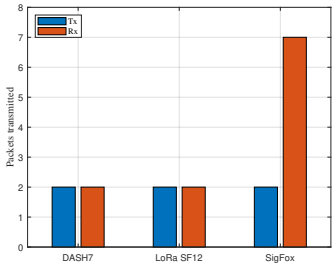
a higher PER, while in other situations the device may benefit from the capture effect when it moves closer to a gateway.

For every PER, the simulation was run 25 times, during which a device will try up to 10 times before stopping the registration process. `MAX_ACK_REQUESTS` is set to 3, therefore a device may try up to 30 times to deliver a fragmented registration message.

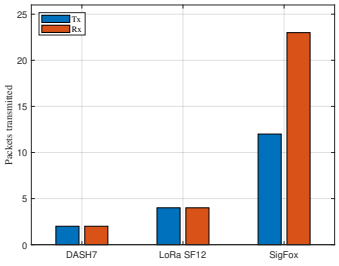
The results for a EU LoRaWAN SF12 device and a Sigfox device are given in Figure 3.14. It can be seen that, due to its longer time on air and smaller payload size, registering a Sigfox device using ND can take in some situations nearly 100 times the time required to register a LoRaWAN device. In such situations, it might be beneficial to use a higher registration back-off period to decrease the network's PER and consequently lower the



(a) SCHC Registration



(b) SCHC Neighbor Discovery

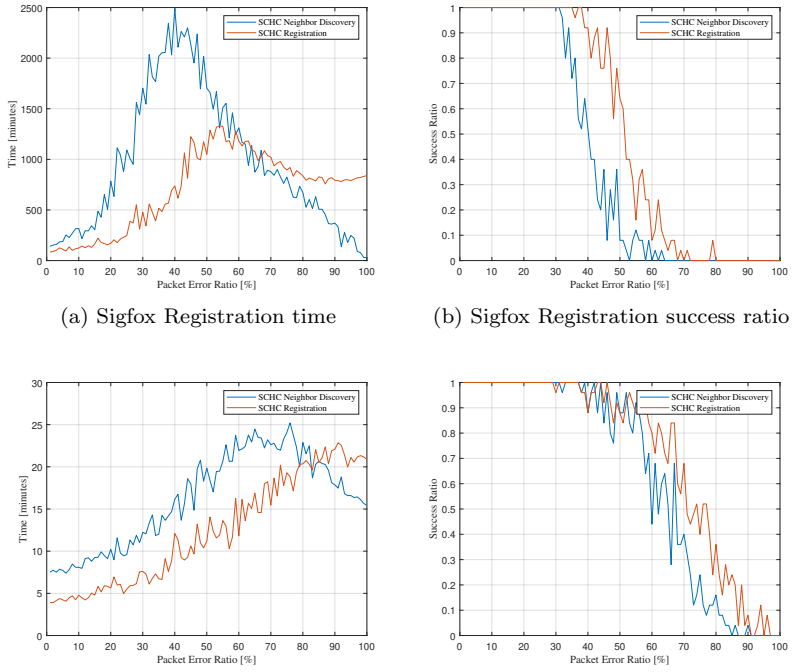


(c) Optimized Neighbor Discovery

Figure 3.13: The total packet overhead for a single registration attempt using both registration mechanisms for different technologies, compared with regular Optimized Neighbor Discovery.

registration time. The graph shows that at least 92 minutes are required to register a Sigfox device to the network using SCHC Registration and 160 minutes are required for SCHC ND in a best case scenario. However, only 4 to 7 minutes are required over a LoRaWAN network. Every registration attempt incorporates the time on air and a 1% duty cycle restriction, which applies in Europe.

Both protocols will impact—and be impacted by—the point at which the network will collapse. We define this point when less than 50% of the registrations succeed. In order to visualize this, the success ratio is given for both protocols and wireless technologies. For the SCHC Neighbor Discovery method, less than 50% of the registrations will succeed for a LoRaWAN SF12 device at a PER of 60%. For a Sigfox device, the capsizing point lies at a PER of 40%. This is again due to the lower available payload size, which requires fragmentation. The SCHC Registration method is impacted less by a higher PER due to the fact that less messages are required, and can therefore cope better with a higher PER than ND. However, in networks



(a) Sigfox Registration time

(b) Sigfox Registration success ratio

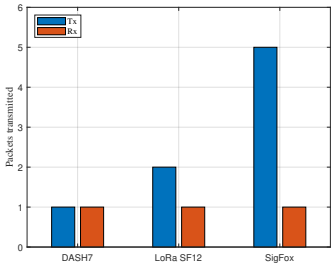
(c) LoRaWAN SF12 Registration time

(d) LoRaWAN SF12 Registration success ratio

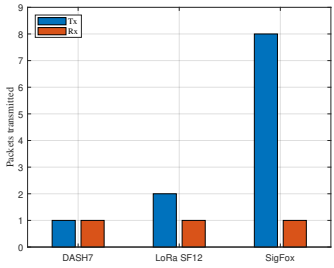
Figure 3.14: The total registration time for different packet error ratios (PERs) for both proposed mechanisms.

with a particularly high PER, the registration time for ND is lower than the SCHC Registration method. Devices have a very low chance of receiving a RA in response to a RS (which has a very low binary overhead) and consequently will not start the NS/NA sequence. This way, the RS can act as a PER indicator and can be used as such to increase the back-off period. In a fragmented environment, the SCHC Registration time decreases after the point at which the network collapses. This is due to the fact that a failed fragmentation sequence will stop the registration; the next fragments are discarded by the device.

However, a device employing the SCHC Neighbor Discovery mechanism, will not be able to communicate over a compressed link without configuring its context. Therefore, the next section evaluates the overhead of the SCHC Control Option.



(a) SCHC IPv6/UDP source and destination (IPv6 and port) context configuration overhead in `ack-on-error` mode



(b) SCHC complete IPv6/UDP rule context configuration overhead in `ack-on-error` mode

Figure 3.15: SCHC ND IPv6/UDP context configuration.

3.6.3 SCHC ND: Context Configuration Overhead

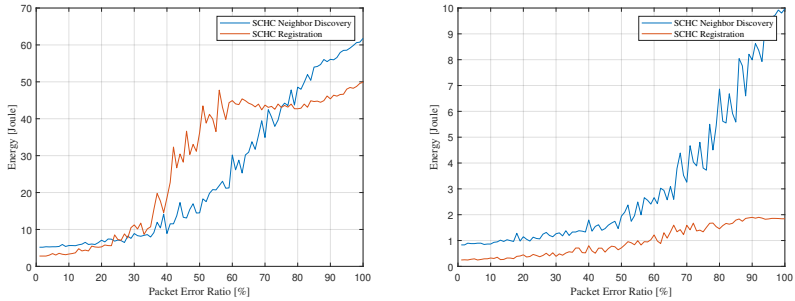
Once a SCHC ND-enabled device has registered to the gateway, it should register its context to optimize the communication flow in terms of header overhead. In order to provide a fair comparison with the SCHC Registration mechanism, the overhead is calculated in a scenario where it is assumed that a rule is installed in the registry that can be updated using the SCHC Control mechanism.

A first example updates the IPv6 source and destination and the UDP source port and destination port. In order to do so, a CAO contained 6 Fixed Size Context Options (IPv6 SRC PRE, IPv6 SRC IID, IPv6 DST PRE, IPv6 DST IID, UDP SRC, and UDP DST). The total overhead boils down to 60 bytes (480 bits), resulting in two packets for a LoRa SF12 device. However, lower MTU-technologies will fragment the context registration request over more packets, illustrated in Figure 3.15a. A single acknowledgment suffices to acknowledge the fragmentation window in case no packet was lost.

For further reference, we also conducted a test what the overhead would be for the configuration of a complete IPv6/UDP rule. Therefore, a complete CAO contained 14 Fixed Size Context Options. Configuring a complete IPv6/UDP context, results in 89 bytes (712 bits) overhead for Sigfox and 83 bytes (664 bits) overhead for a LoRa SF12 device, resulting in 8 and 2 uplink packets, respectively, illustrated in Figure 3.15b.

3.6.3.1 Energy overhead

In exchange for the provided flexibility using the context configuration protocols, a higher energy consumption can be expected. We take the PER



(a) Sigfox context registration overhead

(b) LoRaWAN context registration overhead

Figure 3.16: Energy required to configure the SCHC context for different PERs, for both proposed mechanisms. Note the different scale of the Y-axis.

to measure the extra energy consumption in the situation described in Figure 3.15a, compared to a static context. The results are presented in Figure 3.16. Configuring a context using the ICMPv6 method, will at least require 5 Joules for a Sigfox device while in the best case it will only consume approximately 0.9 Joules on a LoRa SF12 device. Once the PER starts to increase, fragmentation will have a large impact on the energy consumption of both protocols.

From these graphs we can conclude that devices should increase their back-off period or limit the number of attempts to configure a context once the delivery ratio starts to decrease.

3.6.4 LwM2M Configuration

In order to show the relevance of the developed solutions, an example of a device using the widespread LwM2M protocol is evaluated. An overview of the bootstrapping and registration process is given in Figure 3.17. A client is configured with the credentials of the LwM2M bootstrap server, to which it can request the information of the LwM2M server.

The bootstrapping and registration of a device using a static configuration is compared against both proposed solutions. We assume that the address of the LwM2M bootstrap server is configured a priori at the SCHC adaptation layer on both sides of the network. For every test a 6-bit SCHC rule id was used. A 1-byte CoAP token and 2-byte CoAP message id are sent with every message. Some CoAP option fields, such as the LwM2M endpoint name, were left uncompressed in order to provide more flexibility to the upper layer protocols. Figure 3.18 shows the cumulative bit overhead for

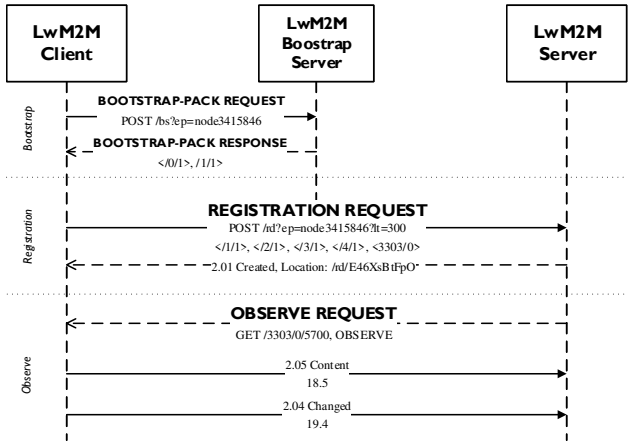
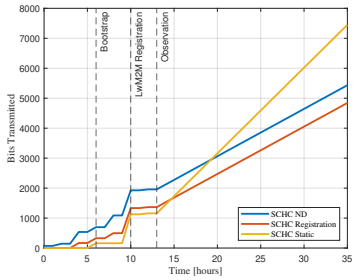


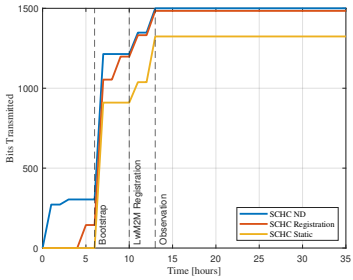
Figure 3.17: LwM2M bootstrap, registration, and observe request.

all configurations for a LoRa SF12 device. While the other configurations wait, SCHC ND will first send the RS/RA and NS/NA before configuring its context. At this point, the SCHC Registration mechanism will also start the registration and context configuration of the bootstrap server. The fifth transmission initiates the LwM2M bootstrap-pack sequence. Until this point, the static configuration clearly outperforms both solutions. After receiving the LwM2M server objects (the 6th transmission in downlink, which explains the large spike), both registration mechanisms again register their context to the SCHC gateway, this time provisioning the location of the LwM2M server. All configurations now register their objects and instances to the LwM2M server. Again, the static configuration outperforms both solutions. However, from this point on, the static configuration will have to transmit the IPv6 address of the LwM2M server as part of the SCHC residue in every message. Once the server starts observing the temperature value of the sensor device, every notification event will also contain this IPv6 address and both registration mechanisms will outperform the static solution after only 20 and 25 transmissions. The inability to configure the IPv6 address of the destination therefore has a large impact on the energy consumption of the device, which can be seen in part (c) of the figure.

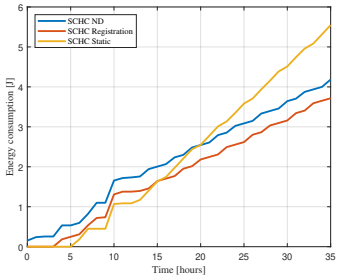
The above shows the relevance of the ability to configure the SCHC adaptation layer: the registration overhead will eventually result in less overhead, more flexibility and consequently less energy consumption. Figure



(a) Cumulative bit overhead in uplink



(b) Cumulative bit overhead in downlink



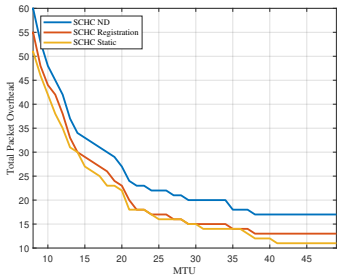
(c) Cumulative energy consumption

Figure 3.18: The cumulative bit and energy overhead for the static and dynamic solutions for a LoRaWAN SF12 device sending an observation notification of its temperature value every hour.

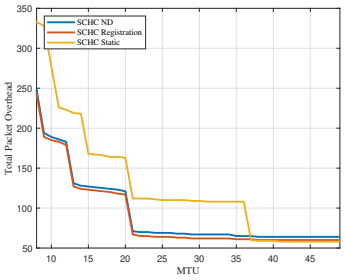
3.19 leverages on this by comparing the total packet overhead for different MTU sizes. Again, an uncompressed LwM2M server address is used for the exchanged packets. Figure 3.19a shows the complete SCHC and LwM2M registration, after which a single observation notification is sent from the device. It can be seen that both registration mechanisms exchange more packets than the static solution. However, the second part of the figure shows the total packet overhead after the registration sequence and 48 observation notifications from the device. It becomes clear that for devices with a low MTU—as they will have to fragment large SCHC residues—it is very useful to configure a header value which has changed during their lifetime.

3.6.5 CoAP Compression

The previous sections showed the significance of both registration mechanisms compared to the current static solution. However, as the SCHC Registration mechanism requires double compression, this will also impact the total bit



(a) Total packet overhead per MTU for 1 notification



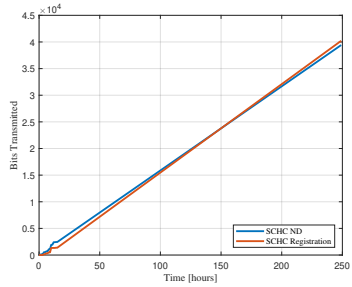
(b) Total packet overhead per MTU for 48 notifications

Figure 3.19: The total packet overhead for the complete LwM2M registration flow and different number of observation notifications.

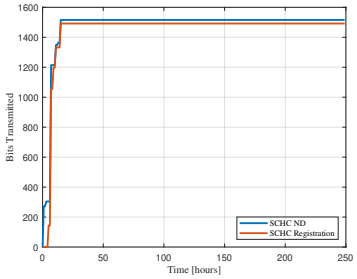
overhead. In this evaluation, we suppose that a CoAP context is deployed a priori at the LwM2M server for the SCHC Registration mechanism. The SCHC ND method, on the other hand, will configure the CoAP context after the LwM2M registration to the SCHC gateway, with an overhead of 488 bits. However, as can be seen from Figure 3.20, after approximately 150 transmissions, the SCHC Registration mechanism will have a higher cumulative bit overhead than the SCHC ND method, due to the double compression, which adds another byte rule id to every packet. We define bit overhead as protocol overhead; bits that the user is not interested in.

3.7 Discussion

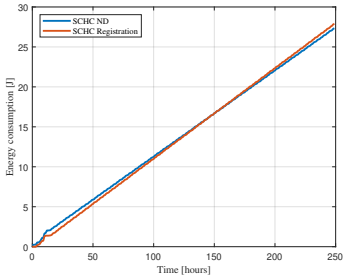
SCHC has been proven to be a valuable tool to bring IPv6 to constrained wide-area networks. However, the static nature of the compression context can have a large impact on the energy consumption. Constrained sensor devices unable to configure the IPv6 address of the server in a LwM2M environment will have to include this value in every packet. After only 25 transmissions, both developed solutions already surpass the energy consumption of the current SOTA. The first solution—the SCHC Registration mechanism—can be used to reduce the protocol overhead to a minimum in changing environments. However, due to the simplicity of the protocol, only four configurable bindings are available per device. We showed that more flexibility can be provided by using an ICMPv6 based mechanism, which however increases the packet overhead, energy overhead, and registration time. Moreover, the ICMPv6 solution can be used in both directions: a device is able to configure the context at the network gateway side and vice



(a) Cumulative bit overhead in uplink



(b) Cumulative bit overhead in downlink



(c) Cumulative energy consumption

Figure 3.20: The cumulative bit and energy overhead for both solutions for a LoRaWAN SF12 device. Note that the SCHC ND mechanism will require less energy in uplink due to the double compression of the SCHC registration mechanism.

versa. This is something that is not supported by the current DNS-based solution from Bernard et al. [8].

Furthermore, the SCHC Registration mechanism does not allow much flexibility and will mainly be used to configure or update a number of generic rules and to register the device to the IPv6 network. For example, employing the more advanced matching operators is only possible if they were set up a priori in the Rule Registry. The ICMPv6-based solution however, can add, update, or remove a rule entry of any kind. In order to keep the SCHC Registration mechanism as simple as possible, and not to interfere with the current specification, it cannot be used to configure the application layer. This forces the end-point and intermediary to implement a SCHC (de-)compression mechanism, which again increases implementation and integration complexity. This, however, is required by the SCHC-CoAP specification to provide end-to-end security and might therefore be used in conjunction with the developed solution. The double compression, however,

adds an overhead to every packet with the size of the rule id, which eventually will result in a higher cumulative bit overhead than the SCHC ND method. Furthermore, the static application layer context, implemented on both endpoints, cannot benefit from the presented context registration mechanism.

Finally, the simulations showed that for both protocols, a mechanism should be incorporated in order to limit the number of registration attempts in networks with a high PER.

3.7.1 CORECONF

A possible alternative for the work presented in this paper could make use of other management solutions such as CORECONF. In such architecture, novel mechanisms to register devices and generate their IPv6 addresses would be required. On top of that, every device must implement CoAP and CORECONF, whereas the SCHC standard defines a generic framework that can be used to compress any kind of protocol. For those reasons, the protocols presented in this paper rather build on the SCHC framework than on an application layer protocol. Nevertheless, the YANG data model, presented by the LPWAN WG, might be used in the future to provide interoperability among SCHC contexts. Once Schema Item Identifierss (SIDs) are defined, the management of contexts using the CBOR representation can be compared with other techniques and management protocols, such as LwM2M and the ones presented in this work.

3.8 Future Work

3.8.1 SCHC

Currently, the SCHC standard does not provide a way of piggybacking acknowledgments with data. Both mechanisms, however (and the specification in general), could benefit from such approach. For example, when a device registers to the SCHC gateway using the `ack-on-error` mode, an acknowledgment will be sent in order to complete the transmission window. Therefore, a downlink packet, containing the IPv6 address of the device is queued for LPWAN technologies that make use of an "uplink-triggered" downlink mode, which opens a receive window only after an uplink. The final packet of the registration will only arrive at the device after another uplink is fired. In request–response scenarios, such as a LwM2M bootstrap request, the bootstrap response will only arrive after another uplink of the device, messing with the LwM2M flow.

3.8.2 ICMPv6

This paper presented the compression of ICMPv6 packets using the SCHC framework, which could be used in the future to make LPWAN devices completely IPv6 compliant. However, in order to be fully compliant with RFC 4443 [24], ICMPv6 traffic should propagate to the end device to ensure proper connectivity for IPv6 devices. However, as LPWANs are characterized by high delays and sleepy end-devices, downlink traffic cannot reach the end-devices directly as it would on the internet and will often get queued. Furthermore, the fact that LPWAN devices and gateways in the unlicensed spectrum have to comply with regulatory limitations and therefore as much airtime as possible should be preserved, the network gateway should respond as much as possible on behalf of the device to handle unwanted ICMP traffic [25]. For example, ICMP informational messages initiated by the sensor device should be compressed and forwarded to the corresponding host on the IPv6 network. The Echo Reply should propagate over the LPWAN to the sensor node. However, an Echo Request intended for the sensor device should be intercepted and replied to by the network gateway.

3.8.3 Security

As the SCHC Control Messages contain sensitive information, security should be looked at into more detail. This can be done similarly to IPv6 Routing Protocol for Low-power and Lossy Networks (RPL), where the higher order bit of the Code field (0x80) can be used to denote whether the message has security enabled [26]. Next, a security field should be added between the base object and the ICMPv6 header to support confidentiality and integrity.

Furthermore, the IETF has identified four types of attacks against Institute of Electrical and Electronics Engineers (IEEE)-identifier-based Interface IDs [27]. The following are of concern in LPWANs:

- Correlation of activities over time: an IPv6 address can be used to correlate the activities of a host for as long as the lifetime of the address.

- Location tracking: since the interface identifier portion of the IPv6 address remains constant, it does provide a way for observers to track the movements of a host

- Device-specific vulnerability exploitation: the embedded IEEE identifier can leak information about the device, which could be used by an attacker to detect and exploit well-known hardware or software issues.

In 6LoWPAN, these issues were solved in an extension to the original RFC, entitled Address-Protected Neighbor Discovery for Low-Power and Lossy Networks [28]. Nodes supporting this extension compute a cryptographic identifier (Crypto-ID), and use it with one or more of their Registered Addresses. The Crypto-ID is a replacement for the MAC address of which the ownership can be demonstrated using cryptographic mechanisms (using a public key). This however, requires either a key exchange mechanism or symmetric keys that have been distributed before deployment. Furthermore, in order to proof its identity, a lot of messaging overhead is introduced and requires a more in-depth study for LPWANs to cope with security issues.

References

[1] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*, 1998. Available from: https://tools.ietf.org/pdf/rfc2460.pdf.

[2] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, 2007. Available from: https://www.rfc-editor.org/rfc/pdfrfc/rfc4944.txt.pdf.

[3] B. Moons, A. Karaagac, J. Haxhibeqiri, E. De Poorter, and J. Hoebeke. *Using SCHC for an optimized protocol stack in multimodal LPWAN solutions*. In WF-IoT2019, the IEEE World Forum on Internet of Things, pages 1–6, 2019. Available from: http://hdl.handle.net/1854/LU-8613162.

[4] ETSI. *Cyber Security for Consumer Internet of Things*, 2019. Available from: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.00.00_20/en_303645v020000a.pdf.

[5] W. Ayoub, F. Nouvel, S. Hmede, A. E. Samhat, M. Mroue, and J.-C. Prevotet. *Implementation of SCHC in NS-3 and Comparison with 6LoWPAN*. In 2019 26th International Conference on Telecommunications (ICT), pages 432–436, Hanoi, Vietnam, April 2019. IEEE. Available from: https://ieeexplore.ieee.org/document/8798782/, doi:10.1109/ICT.2019.8798782.

[6] *libschc: A C implementation of the Static Context Header Compression*, September 2021. original-date: 2019-12-18T12:11:32Z. Available from: https://github.com/imec-idlab/libschc.

[7] W. Ayoub, M. Mroue, A. E. Samhat, F. Nouvel, and J.-C. Prévotet. *SCHC-Based Solution for Roaming in LoRaWAN*. In L. Barolli, P. Hellinckx, and T. Enokido, editors, Advances on Broad-Band Wireless Computing, Communication and Applications, volume 97, pages 162–172. Springer International Publishing, Cham, 2020. Available from: http://link.springer.com/10.1007/978-3-030-33506-9_15, doi:10.1007/978-3-030-33506-9_15.

[8] A. Bernard, S. Balakrichenan, M. Marot, and B. Ampeau. *DNS-based dynamic context resolution for SCHC*. In ICC 2020 - 2020 IEEE International Conference on Communications (ICC), pages 1–6, Dublin, Ireland, June 2020. IEEE. Available from: https://ieeexplore.ieee.org/document/9148910/, doi:10.1109/ICC40277.2020.9148910.

[9] K. Q. Abdelfadeel, V. Cionca, and D. Pesch. *Dynamic Context for Static Context Header compression in LPWANs.* In 2018 14th International Conference on Distributed Computing in Sensor Systems (DCOSS), pages 35–42, June 2018. ISSN: 2325-2944. doi:10.1109/DCOSS.2018.00013.

[10] U. Raza, P. Kulkarni, and M. Sooriyabandara. *Low Power Wide Area Networks: An Overview.* arXiv:1606.07360 [cs], June 2016. arXiv: 1606.07360. Available from: http://arxiv.org/abs/1606.07360.

[11] S. Laboratories. *The Evolution of Wireless Sensor Networks*, 2013. Available from: https://www.silabs.com/documents/public/white-papers/evolution-of-wireless-sensor-networks.pdf.

[12] *The Things Industries launches Global Join Server with a series of device makers to simplify LoRaWAN device provisioning*, January 2020. Library Catalog: thethingsindustries.pr.co. Available from: https://thethingsindustries.pr.co/185845-the-things-industries-launches-global-join-server-with-a-series-of-device-makers-to-simplify-lorawan-device-provisioning.

[13] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. *Neighbor Discovery for IP version 6 (IPv6)*, 2007. Available from: https://tools.ietf.org/pdf/rfc4861.pdf.

[14] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs).* Technical Report RFC6775, RFC Editor, November 2012. Available from: https://www.rfc-editor.org/info/rfc6775, doi:10.17487/rfc6775.

[15] T. Watteyne and P. Thubert. *Efficient 6LoWPAN Neighbor Discovery applied to Multilink IoT subnets.* In 2015 IEEE International Conference on Communications (ICC), pages 642–647, London, June 2015. IEEE. Available from: http://ieeexplore.ieee.org/document/7248394/, doi:10.1109/ICC.2015.7248394.

[16] M. A. Seliem, K. M. Elsayed, and A. Khattab. *Optimized neighbor discovery for 6LoWPANs: Implementation and performance evaluation.* Computer Communications, 112:73–92, November 2017. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0140366417308988, doi:10.1016/j.comcom.2017.08.013.

[17] S. Sinche, D. Raposo, N. Armando, A. Rodrigues, F. Boavida, V. Pereira, and J. S. Silva. *A Survey of IoT Management Protocols and Frameworks*. IEEE Communications Surveys Tutorials, 22(2):1168–1190, 2020. Conference Name: IEEE Communications Surveys Tutorials. doi:10.1109/COMST.2019.2943087.

[18] M. Veillette, P. van der Stok, A. Pelov, A. Bierman, and I. Petrov. *CoAP Management Interface (CORECONF)*, April 2020. Available from: https://tools.ietf.org/pdf/draft-ietf-core-comi-10.pdf.

[19] A. Minaburo and L. Toutain. *Data Model for Static Context Header Compression (SCHC)*, October 2020. Available from: https://tools.ietf.org/pdf/draft-ietf-lpwan-schc-yang-data-model-03.pdf.

[20] O. LwM2M. *Lightweight Machine to Machine Technical Specification: Core*, November 2020. Available from: http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf.

[21] J. Zuniga, C. Gomez, and L. Toutain. *SCHC over Sigfox LPWAN*. October 2020. Available from: https://tools.ietf.org/pdf/draft-ietf-lpwan-schc-over-sigfox-04.pdf.

[22] A. Minaburo, L. Toutain, and R. Andreasen. *LPWAN Static Context Header Compression (SCHC) for CoAP*, December 2019. Available from: https://tools.ietf.org/pdf/draft-ietf-lpwan-coap-static-context-hc-12.pdf.

[23] B. Reynders, Q. Wang, and S. Pollin. *A LoRaWAN module for ns-3: implementation and evaluation*. In Proceedings of the 10th Workshop on ns-3 - WNS3 '18, pages 61–68, Surathkal, India, 2018. ACM Press. Available from: http://dl.acm.org/citation.cfm?doid=3199902.3199913, doi:10.1145/3199902.3199913.

[24] A. Conta, S. Deering, and E. M. Gupta. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, 2006. Available from: https://tools.ietf.org/pdf/rfc4443.pdf.

[25] L. Toutain, D. Dujovne, D. Barthel, J.-C. Zúñiga, and A. Kandasamy. *OAM for LPWAN using Static Context Header Compression (SCHC)*. draft-barthel-oam-schc-00, 2019. Available from: https://tools.ietf.org/html/draft-barthel-oam-schc-00.

[26] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander. *RPL: IPv6 Routing Protocol*

for Low-Power and Lossy Networks. Technical Report RFC6550, RFC Editor, March 2012. Available from: https://www.rfc-editor.org/info/rfc6550, doi:10.17487/rfc6550.

[27] A. Cooper, F. Gont, and D. Thaler. *Security and Privacy Considerations for IPv6 Address Generation Mechanisms*. Request for Comments RFC 7721, Internet Engineering Task Force, March 2016. Num Pages: 18. Available from: https://datatracker.ietf.org/doc/rfc7721, doi:10.17487/RFC7721.

[28] P. Thubert, B. Sarikaya, M. Sethi, and R. Struik. *Address-Protected Neighbor Discovery for Low-Power and Lossy Networks*. Request for Comments RFC 8928, Internet Engineering Task Force, November 2020. Num Pages: 29. Available from: https://datatracker.ietf.org/doc/rfc8928, doi:10.17487/RFC8928.

4

Efficient Vertical Handover in Heterogeneous Low Power Wide Area Networks

In the previous Chapters, the relevance and limitations of Static Context Header Compression (SCHC) were discussed. It was shown that a registration solution is required in a wireless low power network that truly follows the spirit of the Internet Protocol (IP). This Chapter will look at more complex settings and configurations of Low Power Wide Area Networks (LPWANs). Multiple technologies will be combined to cope with the limitations of one another. This Chapter will propose an architecture to combine these networks, since it is not possible to provide network- and device interoperability in heterogeneous LPWANs without a new middleware. Furthermore, an algorithm for vertical handovers in LPWANs is proposed and evaluated. This allows devices to switch between networks that match their needs at a given point in time.

⋆⋆⋆

This chapter is based on the homonymous article by

B. Moons, A. Karaagac, E. De Poorter, and J. Hoebeke

Published in IEEE Internet of Things Journal 7(3), February 2020

Abstract As the Internet of Things (IoT) continues to expand, the need to combine communication technologies to cope with the limitations of one another and to support more diverse requirements, will proceed to increase. Consequently, we started to see IoT devices being equipped with multiple radio technologies to connect to different networks over time. However, the detection of the available radio technologies in an energy efficient way for devices with limited battery capacity and processing power has not yet been investigated. As this is not a straightforward task, a novel approach in such heterogeneous networks is required. This article analyses different LPWAN technologies and how they can be integrated in such a heterogeneous system. Our contributions are threefold: first an optimal protocol stack for a constrained device with access to multiple communication technologies is put forward to hide the underlying complexity for the application layer. Next the architecture to hide the complexity of a heterogeneous network is presented. Finally, it is demonstrated how devices with limited processing power and battery capacity can have access to higher bandwidth networks combined with longer range networks and on top are able to save energy compared to their homogeneous counterparts, by measuring the impact of the novel vertical handover algorithm.

4.1 Introduction

As the IoT continues to grow and, apart from the research community, starts to gain interest for new business use cases, the need for more diverse settings arises. To meet these demands, new LPWAN technologies, such as DASH7, Sigfox and Long Range (LoRa), have entered the market. Many of these communication technologies make use of the unlicensed Industrial, Scientific and Medical (ISM) 915 MHz band (Region 2) or the license-free European Short Range Device (SRD) 863 to 870 MHz band. Due to their unlicensed character, long range and low (energy) cost, they are the perfect candidate for massive low cost sensor deployment.

However, as these technologies are targeting different use cases, their characteristics differ drastically. LoRa and Sigfox, for example, have a throughput of a few hundreds of bits per second but offer in return a range up to 50 kilometers [1]. DASH7, on the other hand, offers a shorter range, but provides throughputs of hundreds of kilobits per second [2]. Also a lot of research has been conducted around DASH7 localization, which can be used to locate an object with a median location error down to 3.9 meters using a single message [3]. Consequently, some technologies are a better choice for low latency and higher bandwidth requirements, while others may be better suited for long range, periodic sensor updates.

Nevertheless, by combining these technologies several issues obstructing large scale IoT adoption may be solved such as Over The Air (OTA) updates and more accurate Global Positioning System (GPS)-less localization for constrained devices. Although not much research has been conducted around this topic, the electronics company Murata recently brought dual mode LoRa/Sigfox modules to the market, supporting other modulation types too (e.g. Gaussian Frequency Shift Keying (GFSK) and On-Off Keying (OOK)) [4]. A single chip is therefore able to switch between different networks with the use of a single antenna.

Such devices however, require to move away from homogeneous to heterogeneous networks where a device can, depending on its current requirements, search for "Always the Best Connectivity" (ABC) [5]. As these technologies currently coexist as vertical silos next to each other, a higher complexity is involved in managing and communicating with such devices and networks. Therefore, multiple problems must be tackled in such configurations, the first and foremost problem being an efficient approach in detecting the presence of and switching to a more capable network. This has been put forward as vertical handover, i.e. the handover between base stations of different wireless technologies, and the handover decision; the selection of the most appropriate wireless network [6].

In this article, the focus relies merely on network detection as this has the largest impact on the energy consumption, which has been determined as the main targeted efficiency.

The first contribution of this paper is consequently a handover algorithm with configurable parameters to provide resilience and to serve a multitude of use cases. The impact of several configurations and the trade-offs amongst these parameters are studied on the basis of simulations. The obtained results indicate that correctly configured devices will consume substantially less energy compared to their homogeneous counterparts as they can take advantage of the complementary characteristics of the different wireless communication technologies they are equipped with. The trade-off between latency, discovery time and reliability becomes clear when the effect of the different parameters is studied. Furthermore, a single protocol stack is presented where the application layer is unaware of the underlying communication technology and a single packet payload structure can be used. The last contribution of this paper is the further elaboration on the architecture of such heterogeneous networks, as presented in previous work [7], to take away the complexity towards application developers.

The remainder of this paper is structured as follows. First, a case study is presented to emphasize the need for heterogeneous LPWA networks. This is followed by the problem statement and research goals, which highlight

the issues faced when developing a low-power network detection algorithm. After Section 4.4 about related work, the network detection algorithm, the architecture and the proposed protocol stack are presented in Section 4.5. Finally, the energy efficiency of the presented algorithm is evaluated to show its flexibility, resilience and how possible extension towards other technologies was taken into account.

4.2 Case study: Construction and Logistics

In order to highlight some of the issues faced when developing applications in heterogeneous LPWANs, a case study is presented in this section. The use case covers a construction and logistics company, where cranes and other material are transported between the construction site and their warehouse. All equipment is used extensively and requires regular maintenance to avoid high damage costs. Currently, such a company might depend on the discipline of their employees to measure the actual usage. This however, is an error prone task which is better solved by measuring the actual usage by equipping the material with accelerometer-enabled devices. Due to the assets' mobility, the trackers are battery powered and must drain as little energy as possible. In order to track the location of the construction tools, the trackers could be equipped with a Global Navigation Satellite System (GNSS). However, such receivers generally consume a lot of power and are therefore not suited for this use case. In order to track the assets, state of the art LPWAN localization techniques can be used by sending regular updates, including accelerometer data, to the back-end over the best available network. Once the asset's usage has reached a certain threshold, maintenance should be notified over one of the available LPWAN technologies. At the construction site and at the warehouse a DASH7 or private Long Range Wide Area Network (LoRaWAN) network is deployed, which allows for OTA updates and accurate low power localization. Since Sigfox ought to provide global coverage the parts in between both sites are covered by this communication technology. An overview of the presented use case is given in Figure 4.1.

4.3 Problem statement and research goals

While some case studies will have devices with a highly predictable trajectory, other cases might encounter objects moving around more randomly without any prior knowledge about the available networks. This requires a resilient algorithm, which incorporates easy, automated configuration for each device and use case. The algorithm should also be able to adapt itself depending on

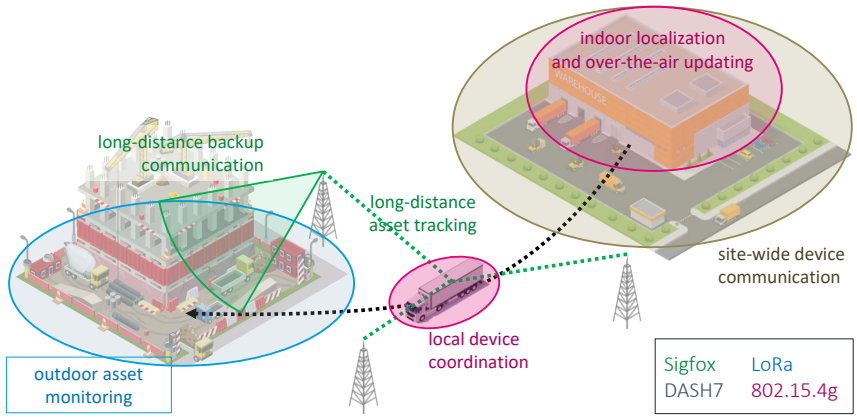


Figure 4.1: The asset tracking use case for a logistics and construction company. Higher throughput, medium range indoor communication can be complemented with lower throughput, long range outdoor communication. Adopted from [8]

any form of input, i.e. the parameters of the algorithm should be modifiable by input from the device itself as well as the back-end.

A second consideration made was that, once a device is connected to a network, it should check at regular intervals whether it is still connected to the network. However, some technologies will impose a lot of restrictions to the device in terms of downlink communication (e.g. Sigfox allows 4 downlink slots of 8 bytes every day). Another reason not to have many downlink slots might be the duty cycle of the gateway (i.e. 0.1%, 1% or 10% [9]). In dense networks, requesting an acknowledgment for each uplink is impractical. Signaling from the back-end to inform the device about future availability of other networks is therefore limited.

A third observation is the capability of the available technologies. Some are more powerful in terms of bandwidth and Maximum Transfer Unit (MTU), whereas others are more focused on long range and limit the other characteristics. It is desirable for the device to switch to a better network once available, since the total time on air and energy consumption should be reduced to a minimum and the reception of large OTA updates made possible. Polling for a better network, however, introduces duty-cycle and energy costs, which imposes restrictions on the polling frequency. The implementation should also not limit the algorithm to the hereafter presented technologies, hence the possibility to integrate new communication technologies should be made easy.

Lastly, a single protocol stack is desirable as the application layer should be unaware of the current underlying technology. A Constrained Application Protocol (CoAP)/User Datagram Protocol (UDP)/Internet Protocol Version 6 (IPv6) approach has been put forward to comply with the internet protocol standards, e.g. IP, UDP, Transmission Control Protocol (TCP), which form the global language spoken on the internet for over 40 years. The major strength of these languages being their maturity and the interoperability between all internet inhabitants. Another benefit is the portability, where different applications can be used independent of the underlying technology. However, some of the available technologies in LPWANs might not be able to transport certain internet protocols and might have to follow a different approach. Sigfox, for example, allows each uplink transmission to use a maximum of 12 bytes, which does not cope with the 40 bytes header overhead of the IPv6 standard.

The goal of this paper is to tackle the heterogeneity of use cases in a multi-modal environment by focusing on a system that can integrate with a multitude of communication technologies for constrained IoT devices. Our research aims to answer the following questions:

1. How can a multi-modal constrained IoT device connect to the best available network for a variety of use cases and still offer enough flexibility?

2. How can low-resource IoT devices use a single stack across a multitude of communication technologies, unaware of the current technology?

3. What is the impact of the presented approach on the energy budget of the device?

4.4 Related Work

Not much research has been conducted around the topic of low power wide area network detection and switching. Though in [10], the authors propose a very high level system architecture of a heterogeneous network consisting of SCHC enabled Narrowband IoT (NB-IoT)/LoRaWAN devices. A machine learning algorithm is proposed at the back-end, which will determine the best communication technology for downlink traffic after sending simultaneously over both networks, resulting in a lot of energy overhead. The architecture however is restricted to LoRaWAN/NB-IoT and neither did the authors perform any form of evaluation.

In [11], the authors propose a central management system for heterogeneous LPWA Networks, where different communication technologies can be

incorporated in a single back-end application. However, heterogeneous devices are not being taken into account, hence no vertical handover algorithm is present. Lastly, the authors in [12] propose a mathematical model for location-based network discovery. The proposed method requires the device to know the location of base stations and an accurate estimate of its own location, which is not always feasible.

4.5 Low Power WAN Discovery

This section will leverage on previous work [7] to enable seamless handovers in LPWANs. After the network architecture is presented, the algorithm is explained, both on back-end side and constrained node side.

4.5.1 Overall System Architecture

In order to take away the complexity of a heterogeneous network, a novel system architecture is required. As proposed in [7] a modular Virtual Network Operator (VNO) is used, where any type of network can be plugged in by using adapters. In Figure 4.2, a high-level overview of the adapter-based architecture is shown.

Once a new network is deployed or being used, a suitable adapter will be installed by the VNO in order to interface with the network infrastructure, e.g. the SigFox cloud, a LoRaWAN Network Server, etc. Towards the right, all adapters have a unified data format, removing the complexity of the underlying network architecture.

The low resource multi-modal IoT devices, shown on the left, use a uniform CoAP/IPv6/SCHC stack across multiple technologies. For each of these devices, the VNO employs an entry in the dictionary where its SCHC rules are exposed. In uplink, end-devices will send compressed SCHC packets, which eventually end up at the LPWAN operator component that will perform decompression. A localization engine is informed about incoming data in order to perform low power localization, based on the received wireless signal. In downlink, clients can generate CoAP/IPv6 packets, which will end up at the VNO that will compress the CoAP/IPv6 packets and forward them to the correct physical device. The compressed packet is forwarded to the corresponding adapter and sent over the active technology.

To realize a working network selection algorithm, able to adapt to the current technology and conditions of the network (e.g. dense networks where the back-end can provide information about less occupied channels) and providing enough flexibility, both the end-devices and VNO are extended with extra intelligence. These extensions and the design vertical handover

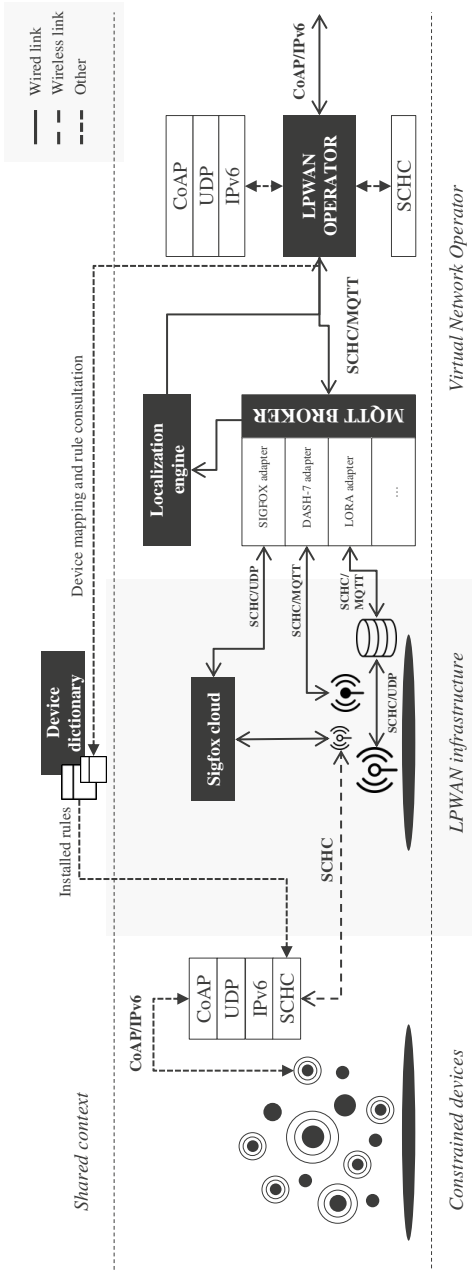


Figure 4.2: The modular architecture of the Virtual Network Operator (VNO) allows easy addition of technologies

algorithm are presented in the following two subsections.

4.5.2 Network Discovery Method

The easiest method for a wireless device to discover reachable wireless networks is by keeping all interfaces on all the time. Another way of discovering a wireless network might happen by actively scanning a channel by sending Probe Requests and waiting for responses, such as used in the IEEE 802.11 scanning phase [13]. Also, as proposed in [6] the position information of the device and a Location-Service Server can assist the device to efficiently discover and connect to the best available network.

These methods, however, are very energy consuming and are impractical for some LPWAN technologies as they also have to take duty cycle restrictions into account. Therefore, an LPWAN tailored algorithm is required which makes a trade-off between power efficiency and network discovery time. Since some use cases require more or less power efficiency and/or network discovery time than others, these requirements should be adjustable by taste. The core of the algorithm is therefore built up around 4 configurable parameters per technology which directly affect these requirements:

- `polling_threshold`: this threshold is used to check if a better network is available
- `downlink_threshold`: this threshold is used to check if the current network is still available
- `max_downlink_retries`: this parameter is used to indicate the number of retries once a downlink is requested
- `priority`: this parameter is used to indicate which technology gets priority when 2 or more technologies overlap in time

The unit of the first two thresholds can be either a time period or number of messages transmitted by the end device (e.g. for end-devices with periodic uplink transmissions). The latter unit will be used in the remainder of the paper. An example configuration is given in Table 4.1.

This configuration means that when the constrained node is not connected to LoRaWAN it will check every 10 messages if such a network is available. If the device is currently connected to a LoRaWAN network, it will expect an acknowledgment every 4 messages. This logic is realized by means of a state machine, as is explained in the following subsection.

Table 4.1: Network driver thresholds. `polling` and `downlink` can either be a time period or a number of transmitted messages. `retries` is a regular counter. `priority` indicates the technology capacity. A higher number means a higher throughput.

Technology	`polling`	`downlink`	`retries`	`priority`
Sigfox	-	35	0	0
LoRaWAN	10	4	1	1
DASH7	6	2	2	2

4.5.2.1 End-device state machine

Figure 4.3 depicts how the logic for the constrained device is separated in 2 layers: the application layer and the communication layer.

The application layer takes care of CoAP requests and responses and processes input data from the sensors, while the communication layer keeps track of the current network interface and implements the logic to switch between the different interfaces. Once a CoAP request is pending at the application layer, it will pass the message to the communication layer. From there the following flow applies:

1. if Quality of Service (QoS) is required by the application, a different approach may be requested, otherwise, every message passed to the communication layer will increment a counter, which is used to check if it matches with one of the 2 parameter thresholds of a technology.

2. the message counter is checked against all network interfaces' `polling_threshold` parameter. Once $counter \mod polling\_threshold$ is 0, the network interface is added to the list to poll for, otherwise

3. the counter is matched against the **current network driver** its `downlink_threshold` parameter. If the mod operation returns 1, a regular uplink will be sent over the current technology. Otherwise the list of network drivers to poll for

4. is run over for `max_downlink_tries` times, while requesting an acknowledgment from the back-end. If no response is received, the device will return to the technology previously connected to.

5. in order to meet the regulatory limitations of duty cycle limited technologies, the message will first pass the duty cycle check component, which keeps track of the time on air of every duty cycle limited transmission. If the current technology is bound to a duty cycle, then the

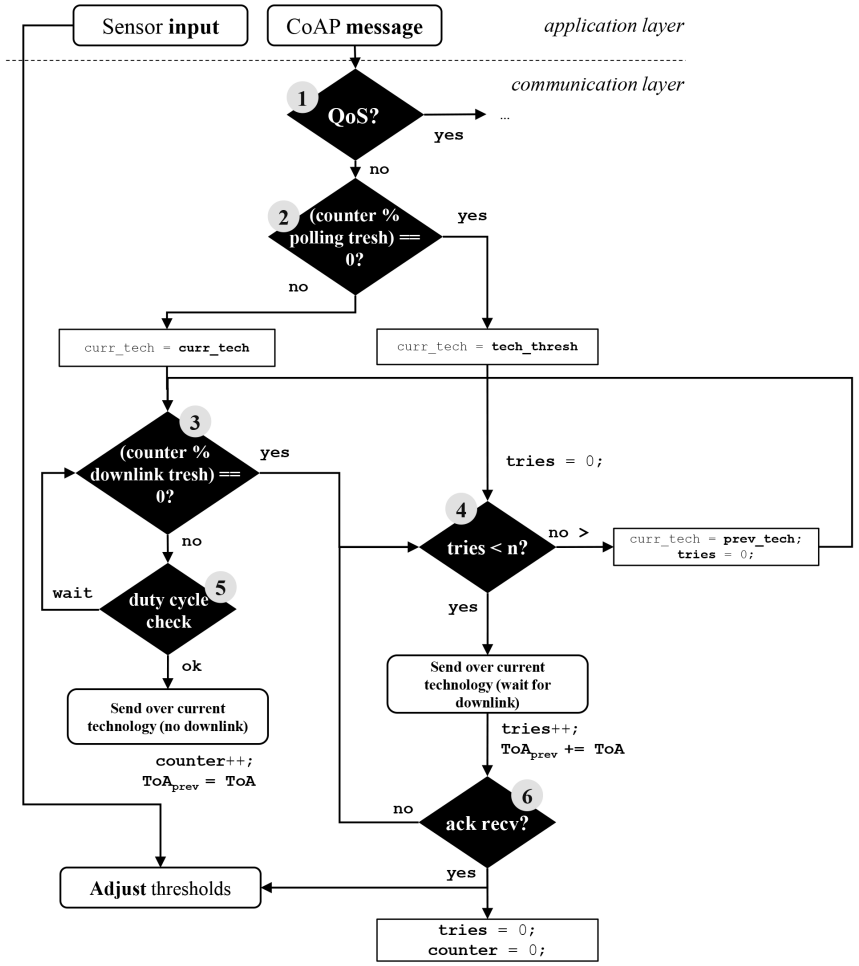


Figure 4.3: The state machine for Low Power WAN detection of the constrained device.

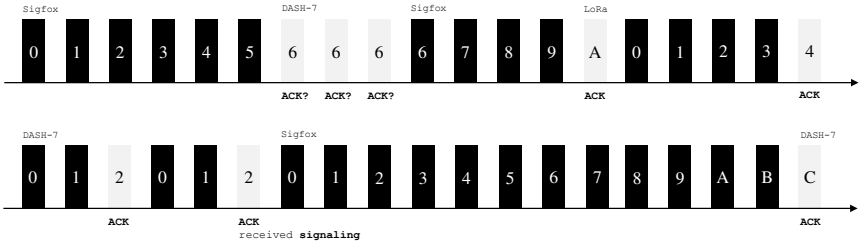


Figure 4.4: Switching example and signaling from the back-end to the device

next transmission will be scheduled following Equation 4.1

$$t_{wait} = (100 - DC) \cdot ToA_{prev} \tag{4.1}$$

with DC, the duty cycle in percent and ToA the previous (cumulated) time on air. When requesting downlink information, the duty cycle is not taken into account to ensure fast handovers, but cumulated for the next uplink-only transmission. This is in line with the ERC/REC 70-03 regulations, as long as the total time on air is calculated based on a one hour period. [14]

6. Finally, if an acknowledgment is received, 2 things are indicated:
   (a) the current technology is available
   (b) the contents of the acknowledgment may be used to update the parameters of the threshold

4.5.2.2 Network Drivers

Information for each physical interface is kept in a `network_driver` instance (Listing 4.1) as part of a linked list, derived from the Light Weight IP (lwIP) implementation [15].

A network driver will forward data to the correct network interface through the `send` pointer and can be initialized and stopped using the appropriate function pointers. Each network driver also implements the parameters required by the algorithm as discussed in the previous section. All interfaces are kept in a linked list, which makes it easy to loop over and add or remove any.

Listing 4.1: `network_driver` structure

```
struct network_driver {
struct network_driver* next;
char name[4];
uint16_t mtu;
uint8_t polling_threshold;
uint8_t downlink_threshold;
uint8_t max_downlink_retries;
uint8_t priority;
void (*init)(void);
void (*stop)(void);
uint8_t (*send)(uint8_t* buf, uint16_t len);
}
```

4.5.2.3 Example

As table 4.1 indicates, the algorithm will check every six unacknowledged messages if any DASH7 network is available. The upper part of Figure 4.4 demonstrates the initial configuration, where the device is connected to the Sigfox network and fails to discover a DASH7 network after its `polling_threshold` (6) and its number of `downlink_retries` (2). Sigfox is used as a fall back, as we assume this is always available. Four messages later, the device reaches the `polling_threshold` for LoRaWAN and receives an acknowledgment over this technology. Once a new network is discovered, the message counter is reset and the device will start to communicate over this network. Once the message counter reaches the technology's `downlink_threshold`, the device will ask for a signaling message from the back-end to check if the network is still available.

The second example shows a device connected to a DASH7 network with the `downlink_threshold` parameter set to 2. This means the DASH7 device will ask for a downlink every 2 uplink messages in order to test the network availability. As the back-end has information about the future network availability it signals the device to use Sigfox for the coming hours. By adjusting the thresholds of the other technologies, the device won't poll for the availability of other technologies, preserving energy and flexibility.

4.5.3 Virtual Network Operator

The back-end implements a routing table, keeping track of the IPv6 address(es) of a single device with multiple Extended Unique Identifiers (EUI) as well as the active technology. An outgoing IPv6 packet, i.e. an IPv6 packet going to the LPWA network, is matched against a device and forwarded to the device over the last active technology. However, the low power nature, i.e. the ability to receive data after an uplink packet, of the network requires the back-end to keep track of outgoing packets as shown in Figure 4.5.

Upon a new request from the IPv6 network, the IPv6/UDP/CoAP headers are compressed using the SCHC adaptation layer. Next, the active technology is checked. Some configurations, e.g. LoRaWAN Class C, allow instantaneous downlink communication, for others the packets are sequentially added to the queue. Each request coming from the LPWA network will trigger the VNO to match the EUI of a particular technology to a device, update the active technology, check if there are any packets present for that device in the queue and forward the first added packet (FIFO).

Once a constrained device sends a request to the back-end system, asking for confirmation about the availability of the network, updated parameters

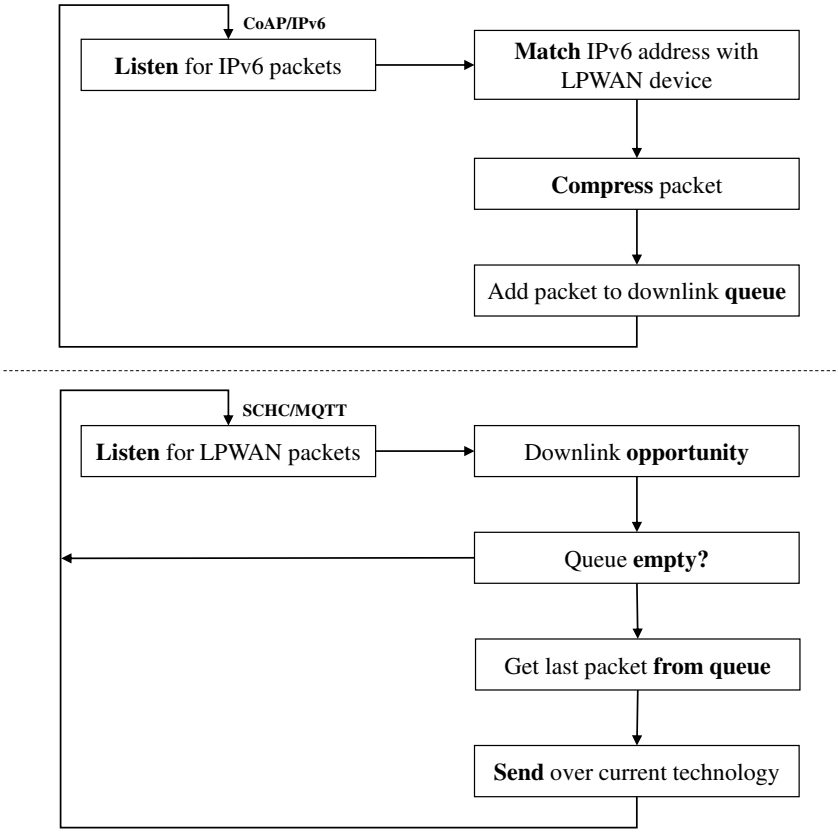


Figure 4.5: The main loop in the back-end. Incoming SCHC packets trigger transmission of queued compressed IPv6 packets.

of the `polling_threshold` and `downlink_threshold` can be piggybacked on the network status acknowledgment.

4.6 Performance Evaluation

4.6.1 Energy Overhead

As LPWAN devices are required to operate on a single battery charge for multiple years, energy consumption is one of the prime criteria in evaluating the feasibility of new concepts and implementations. While the presented handover schema may have many benefits (the most important one being the higher available bandwidth and bit rate), it may also introduce some

undesirable downsides, such as extra energy consumption. Therefore, a simulation was performed in Matlab in order to evaluate the feasibility of the proposed algorithm.

4.6.1.1 Energy Model

In order to evaluate the proposed algorithm based on energy consumption, header overhead, MTU, current per transmission and receive power and data rate were gathered from datasheets and are summarized in table 4.2.

As the energy consumption is mainly determined by the time on air of a device, and the time on air is depending on the bit rate and the physical layer packet size, the following equation can be used:

$$J_{tx} = \frac{(PL + H) \cdot 8}{R_{tech}} \cdot I_{tech} \cdot V \tag{4.2}$$

Where PL and H are respectively the payload length and header length in bytes, divided by the data rate in bps. H incorporates the complete physical (including ramp-up, preamble and ramp-down) and Medium Access Control (MAC) layer headers. the uplink frame This equation does not apply to LoRa as the proprietary Chirp Spread Spectrum (CSS) modulation is used there. Nevertheless, Semtech does provide ways to calculate the data rate and time on air for LoRa enabled devices [16].

First of all, in order to know the time on air, the total number of payload symbols must be calculated using the following formula:

$$n = 8 + \left\lceil \frac{8PL - 4SF + 16CRC - 20EH}{4(SF - 2DE)} \right\rceil \cdot (CR + 4) \tag{4.3}$$

With EH and CRC being the presence of an explicit header or cyclic redundancy check respectively (1 or 0) and DE low data rate optimization.

Once the total amount of symbols are calculated, also the time (in ms) required to transmit one symbol (equation 4.4) must be known.

$$t_{sym} = \frac{2^{SF}}{BW} \tag{4.4}$$

Furthermore, as every frame starts with a preamble with a configurable length (l), this must be calculated too (equation 4.5).

$$t_{pr} = (l + 4.25) \cdot t_{sym} \tag{4.5}$$

Finally, the previous equations can be combined when calculating the energy for a LoRa enabled device (equation 4.6), divided by 1000 to convert from ms to s.

$$J_{tx} = \frac{I_{tx}(n \cdot t_{sym} + t_{pr}) + 2I_{rx} \cdot t_{pr}}{1000} \cdot V \tag{4.6}$$

Table 4.2: LPWAN overview of data-rate, average transmit and receive power requirements, Physical layer header size, MAC layer header size and MTU

Technology	Configuration	R (kbit/s)	I_{rx} (A)	I_{tx} (A)	PHY (B)	MAC (B)	MTU (B)
Sigfox	Uplink	0.1	-	0.030	6	8	12
	Downlink	0.6	0.01	-	18	2	8
LoRa	SF7 CR 4/5 BW 125 kHz	5.4688	0.0108	0.029	-	13 - 28	222 - 207
	SF8 CR 4/5 BW 125 kHz	3.125	0.0108	0.029	-	13 - 28	222 - 207
	SF9 CR 4/5 BW 125 kHz	1.7578	0.0108	0.029	-	13 - 28	115 - 100
	SF10 CR 4/5 BW 125 kHz	0.9766	0.0108	0.029	-	13 - 28	51 - 31
	SF11 CR 4/5 BW 125 kHz	0.5371	0.0108	0.029	-	13 - 28	51 - 31
	SF12 CR 4/5 BW 125 kHz	0.293	0.0108	0.029	-	13 - 28	51 - 31
DASH7	Lo-Rate PN9 w/ FEC	4.8	0.0108	0.029	8	5 - 13	251
	Normal PN9 w/ FEC	27.7775	0.0108	0.029	8	5 - 13	251
	Hi-Rate PN9 w/ FEC	83.3335	0.0108	0.029	10	5 - 13	251

Since a LoRaWAN class A device will open 1 or 2 receive windows for possible downlink communication, which is at least equal to the time required to detect a preamble (t_{pr}), this is included in the model.

In order to show the energy efficiency for each technology, the energy overhead per 12 bytes uplink and 8 bytes downlink is shown in Figure 4.6. Sigfox tends to have the largest energy consumption, since 12 bytes in uplink require 2.08 seconds of airtime. Due to a data rate twice as low for downlink communication, LoRa with SF12 will consume almost twice the energy of an 8 byte Sigfox packet. 1 receive window for LoRa is assumed for downlink communication. The graph clearly shows the low energy overhead of DASH7.

Based on this, it can be concluded that it would be interesting to benefit from the lower energy consumption and higher data rate offered by DASH7 while still obtaining the range Sigfox has to offer.

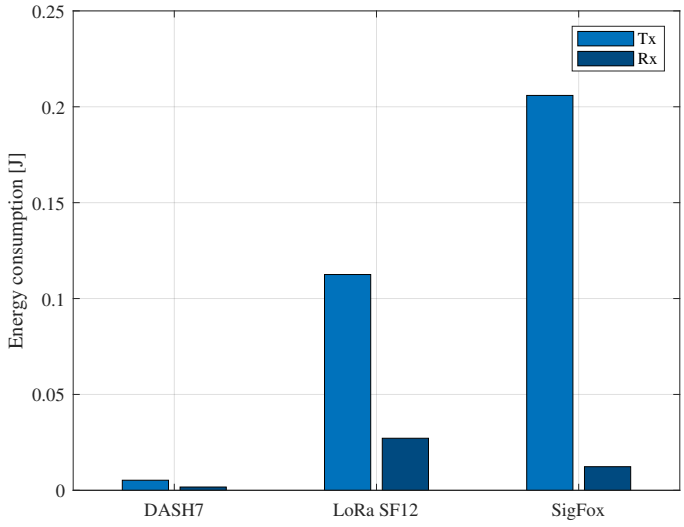


Figure 4.6: The energy overhead for each technology for 12 bytes in uplink and 8 bytes in downlink conform the presented models

In the next section, the benefit and overhead of the proposed handover algorithm is evaluated using a simulation in Matlab.

4.6.1.2 Simulation

In this section, the algorithm will be evaluated theoretically to determine the energy consumption overhead. The energy overhead calculations of

Table 4.3: Network availability probability (p)

Technology	Worst Case	Bad Case	Medium Case	Best Case
Sigfox	1	1	1	1
LoRaWAN	0	0.25	0.5	1
DASH7	0	0.25	0.5	0.75

Table 4.4: Median energy consumption (in Joule) for a heterogeneous device for different cases over a 24 hour time span with N being the total consecutive messages over the same technology before disconnection.

N	Worst Case	Bad Case	Medium Case	Best Case
2	32.16	24.89	20.68	16.41
5	32.16	20.94	15.18	10.75
10	32.16	17.17	11.41	7.51

a heterogeneous LoRaWAN - DASH7 - Sigfox device based on Table 4.2 combined with the network driver principle of Section 4.5.2.2 are implemented in Matlab. The simulation also implements the state machine of Section 4.5.2 and calculates, based on the network availability, the energy overhead of every transmitted message and possible downlink traffic.

As the availability of the networks cannot be modeled based on an exact model, a probabilistic model is maintained, based on a standard normal probability distribution object (p). In our model, 4 cases are put forward: *Worst Case*, where the algorithm is running, but no other networks are available. *Bad Case*, *Medium Case* and *Best Case* with increasing network availability of the other networks. These values are shown in table 4.3.

However, once a device is connected to a network, the probability of sending over that same network increases as the device is still in the presence of the network. Therefore, the variable N is defined, expressing the amount of consecutive messages over the same technology before disconnection.

Every simulation, the corresponding probabilities are applied when the device attempts to send a message. As an example, a device with an uplink transmission frequency of 10 minutes over a 24 hour timespan simulation was run. The values in Table 4.4 indicate how a higher N and p result in a lower energy consumption. Only for a *Worst Case* scenario, the energy budget of the device is not influenced by N, as the only available network is Sigfox.

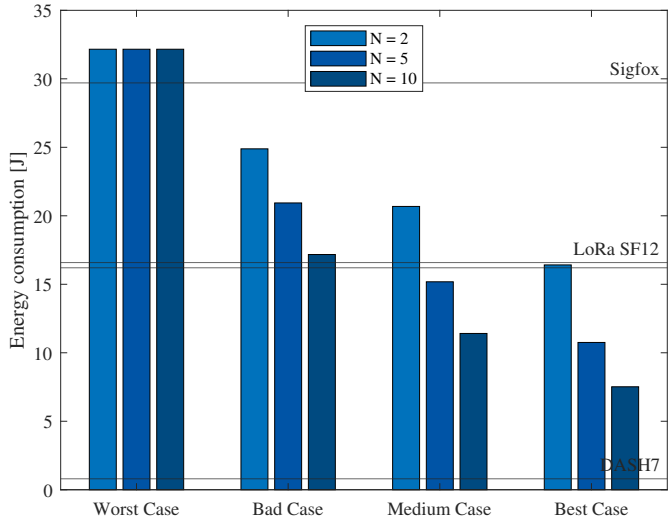


Figure 4.7: Heterogeneous devices using the vertical handover algorithm vs homogeneous devices. Both sending 12 bytes every 10 minutes over a timespan of 24 hours. N represents the total consecutive messages over the same technology before disconnection.

In order to evaluate the energy overhead of the handover algorithm, a comparison is made between single technology devices and a multimodal device. The single technology devices (Sigfox, LoRaWAN SF12 and DASH7) transmit an uplink packet every 10 minutes and receive downlink communication based on the values of Table 4.1 (i.e. Sigfox 35, LoRaWAN 10, DASH7 6). The heterogeneous devices wield the same uplink and downlink frequency as indicated in Table 4.1. In Figure 4.7, the energy consumption of the **single technology devices** over a 24 hour timespan is indicated by the **horizontal lines**. The heterogeneous devices, on the other hand, have been modeled for the 12 different cases from table 4.4 and are illustrated with bars for every N.

It is clear from the Figure that in a *Worst Case* scenario, the heterogeneous device, configured using the parameters from Table 4.1, only consumes 8% more energy than a single technology Sigfox device. However, once the probability of connecting to a better network increases, the multi-modal device quickly outperforms the homogeneous device. For a *Bad Case*, where the heterogeneous device has only 25% chance of connecting to a LoRaWAN or DASH7 network, with probability $N = 10$, it will already perform similarly

to single technology LoRaWAN (SF12) devices.

Once the device is in the proximity of a DASH7 network, a higher data rate is available, hence a shorter time on air. As a result, OTA updates and data offloading can be conducted over this medium range technology. However, once the device starts moving away from the network, which has a range from 1 to 5 km, it may benefit from a lower data rate, longer range technology such as Sigfox or LoRa.

Now that it has been shown that deploying a multi-modal network is beneficial for LPWAN devices in terms of energy consumption and available bandwidth, the next section will study the effect of different parameter configurations in order to find an optimal point of operation.

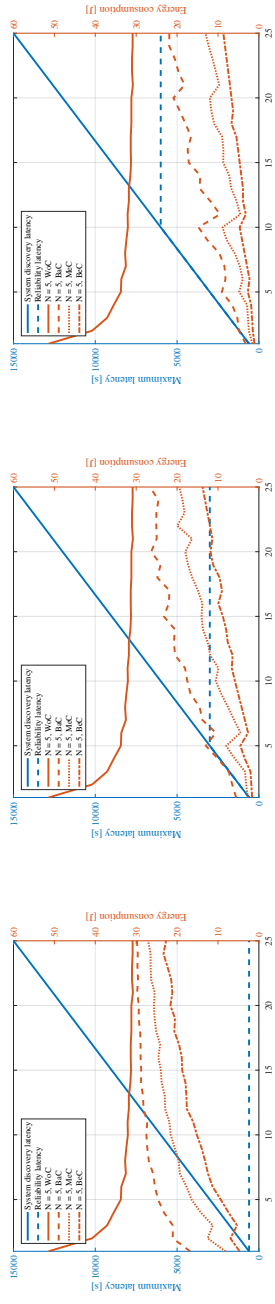
4.6.2 Network Discovery Time and Reliability

Due to the battery powered nature of the targeted LPWAN devices, an optimal point should be chosen where the device consumes the least possible energy, nevertheless an acceptable latency is maintained. In this sense, two types of latency can be distinguished: (1) the *network discovery time*, which is the maximum time before a better network is detected and (2) the *reliability latency*, which is the time before the device will notice a disconnection from the current network. While the polling threshold will directly impact the network discovery time, both the polling and downlink thresholds may affect the reliability latency.

4.6.2.1 Network Discovery Time

Figure 4.8 indicates how a lower polling threshold, i.e. the inter-technology handover check, does not necessarily mean that more energy will be consumed. For the worst case scenario, the heterogeneous device is never able to connect to a lower energy, higher bandwidth network, which will result in a higher energy consumption. However, the higher the probability that the device succeeds in connecting to such networks, the advantage of these low energy networks comes through.

Therefore, once a device enters an area where a higher bandwidth network is available, it should connect to it as fast as possible to make use of the improved characteristics of the network. The maximum latency (in **seconds**) with which it may discover a higher bandwidth network is the maximum *network discovery latency* (S_{max}). The *network discovery latency* depends on the `polling_threshold` (pt) of that technology, as this determines after how many consecutive messages a network can be discovered, and the transmission frequency (f) in Hz and can be modeled as:



(a) Downlink threshold = 1 (b) Downlink threshold = 5 (c) Downlink threshold = 10

Figure 4.8: The energy consumption, network discovery time and reliability latency are affected by varying the threshold parameters. Requesting a downlink after every uplink will decrease the network discovery time, however, with a drastic increase in energy consumption.

$$S_{max} = \frac{1}{f} \cdot pt \tag{4.7}$$

4.6.2.2 Reliability Latency

As a downlink from a discovered network also improves reliability (i.e. piggybacked information about previous transmissions), both thresholds (*pt* and `downlink_threshold` (*dt*)) should be taken into account while modeling the minimum *reliability latency* (L_{min}).

$$L_{min} = \frac{pt + dt - |pt - dt|}{2f} \tag{4.8}$$

However, if the device is not able to discover another network, the maximum *reliability latency* (L_{max}) is equal to equation 4.9.

$$L_{max} = \frac{1}{f} \cdot dt \tag{4.9}$$

The maximum *reliability latency* and *network discovery latency* are modeled in Figure 4.8 for various configurations of the threshold parameters. It can be seen that the *reliability latency* is equal to the threshold with the lowest value until the `polling_threshold` exceeds the `downlink_threshold`. From then on the latency will be equal to L_{max}. On the contrary, it is clear how a higher `downlink_threshold` will restrict the device in fast network discovery, as *pt* does not have an impact on the maximum *reliability latency*.

The *network discovery latency* is affected directly by the `polling_threshold` only, as for every configuration a constant sending frequency of 10 minutes has been used.

Furthermore, it can be seen that the energy consumption of a device in all cases decreases when increasing the *dt* parameter. The constrained node will fire more uplink transmissions without asking for downlink confirmation, which will reduce the time on air, nevertheless with an increasing reliability latency as a consequence.

Another observation made is that an increasing *pt* forces the device to switch between their different network interfaces less frequently, however increases the energy consumption. This can be explained that by increasing the frequency with which the device switches between network interfaces, the probability of finding a higher bandwidth network increases.

Lastly, a reverse curve can be noticed in the *Worst Case* scenario due to the fact that the device will try to detect other networks less frequently. In contrast to the previous observation, no probability influences the underlying network technology, which directly impacts the energy consumption.

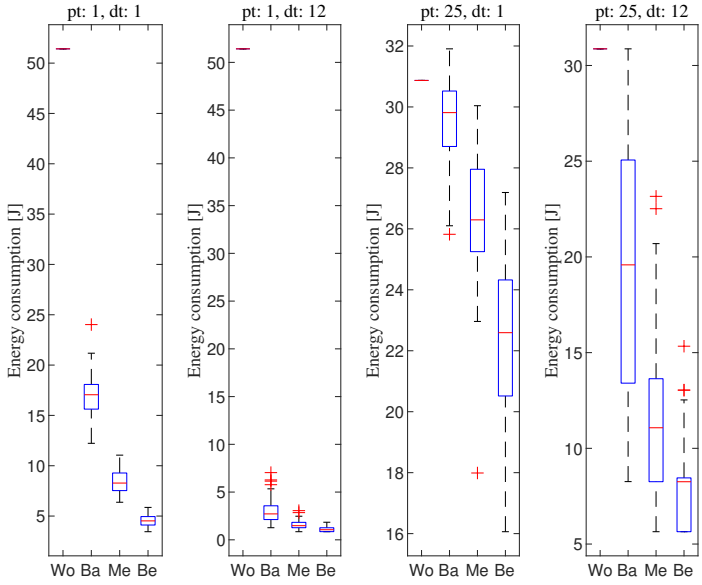


Figure 4.9: Comparison between different configurations for the vertical handover algorithm with N set to 5 for 24 hours

Furthermore, the curve does not change by increasing the *dt* parameter, since the device will remain on the Sigfox network throughout the whole simulation.

4.6.3 Configuration

From the previous section, it may be concluded that the different parameters have a large impact on the latency and energy consumption. Therefore, to have a better understanding when configuring the parameters of the algorithm, 4 extreme configurations were simulated 100 times with their distribution shown in Figure 4.9.

This can be broken down as follows:

- $pt : 1, dt : 1$, the most efficient configuration, i.e. the configuration with the lowest *reliability latency* and the fastest *network discovery time*. The device will poll for a better network and will ask confirmation about the current network after every uplink transmission. This however, is impractical in large networks due to the gateways' duty cycle restrictions.

- $pt : 1, dt : 12$, the multi-modal device will only ask for acknowledgments after 12 uplink transmissions, and may, in a worst case scenario, only notice a disconnection from the network after 12 uplink transmissions. However, the probability of connecting to a better network is high, which increases the *reliability latency*.
- $pt : 25, dt : 1$, the device will notice immediate disconnection from the network, however, once connected to the network with the highest energy consumption and lowest data rate, it will only try to discover a higher bandwidth network after 25 uplink transmissions, which affects the distribution across the different cases significantly.
- $pt : 25, dt : 12$, the median energy consumption decreases, since fewer downlink transmissions are requested, however, the distribution increases due to the increased `downlink_threshold`, which will result in less stable network detection.

From this, it can be concluded that by using the most efficient (in terms of latency) configuration (i.e. $pt : 1, dt : 1$), energy might be saved when there is a high probability of connecting to a lower energy hungry network. However, due to duty cycle regulations, it is not possible for gateways to respond to each message of every device. Therefore, the threshold parameters should be set as high as possible. Nevertheless, this decreases the *reliability latency* and increases the *network discovery time*. Hence, certain device conditions require different configurations, which might be solved by signaling updated parameter thresholds to the device or update the threshold configuration based on sensor information.

4.6.4 Reliability overhead

Some use cases require reliable communication and can not afford missing a maximum of $f \cdot L_{max}$ messages. In order to achieve reliable communication, worst case, the total number of unacknowledged messages that have the possibility to be lost, must be retransmitted using Sigfox. Therefore, as depicted in Figure 4.10, the maximum overhead when adding reliability is the total number of unacknowledged messages multiplied by the energy required for an uplink using Sigfox. When compared to Figure 4.9 it is clear that when reliability is required, a higher dt has a higher probability in consuming less energy.

4.6.5 Implementation Overhead

As constrained devices have only a limited amount of memory ($<$ 50kB Random Access Memory (RAM) and $<$ 256kB of Read Only Memory (ROM)),

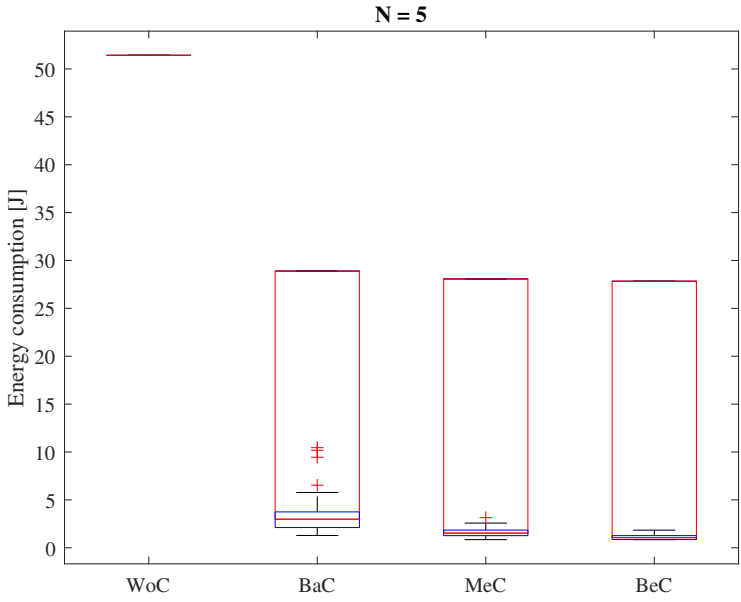


Figure 4.10: The maximum energy overhead while safeguarding reliability

it is important to keep memory consumption to a minimum. Therefore the different components of the proposed system are presented in Table 4.5 with their corresponding overhead. The values were listed and processed using the ARM `readelf` tool, which performs a similar function to `objdump`.

It can be seen from the table that the Low Power Handover (LPH) algorithm requires 538 bytes of RAM, mainly required for message buffers and to keep track of the different network drivers, as explained in Section 4.5.2.2. The state machine itself requires almost 2 kB. In order to have a fully working protocol stack, an adapted version of Adam Dunkels' μIP library is used, where TCP support can be disabled. This adds 1622 bytes of ROM and 409 bytes of RAM. Furthermore, the *p*CoAP library has been integrated to achieve minimal CoAP support and to (de)compress SCHC CoAP packets, resulting in 1858 kB of ROM. Only a limited amount of memory is required for the CoAP stack, since this implementation does not contain more complex CoAP mechanisms.

It can be seen that the implementation overhead of the different components is kept to a minimum in order to target constrained devices.

Table 4.5: Code space required for the different components

	LPH	*p*CoAP	μIP	SCHC	**Total**
RAM	538 B	300 B	409 B	1589 B	2836 B
ROM	1968 B	1858 B	1622 B	16030 B	21478 B

4.7 Conclusion

In this article, a novel network detection algorithm for selecting the best available LPWA technology in a heterogeneous network was presented. To the best of our knowledge, no algorithms are currently available for LPWAN network detection and a novel algorithm and architecture was therefore presented.

In order to have a single stack over multiple technologies, SCHC was presented and used as an adaptation layer below a CoAP/UDP/IPv6 stack in order to enable a standard based, internet compliant Internet of Things. We showed that some technologies, however provide a full vertical stack, resulting in a lot of overhead.

Next, in our evaluation it comes forward that a multi-modal device and the presented algorithm will help to save energy using a correct configuration compared to a homogeneous Sigfox device and some LoRaWAN configurations. Apart from these energy savings, we enable OTA updates for a Sigfox device, which by itself is only capable of receiving 32 bytes every day.

Lastly, it is shown that the total implementation overhead is kept to a minimum in order to cope with constrained IoT device requirements.

References

[1] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer. *A comparative study of LPWAN technologies for large-scale IoT deployment.* ICT Express, 5(1):1 – 7, 2019. Available from: http://www.sciencedirect.com/science/article/pii/S2405959517302953, doi:https://doi.org/10.1016/j.icte.2017.12.005.

[2] M. Weyn, G. Ergeerts, R. Berkvens, B. Wojciechowski, and Y. Tabakov. *DASH7 alliance protocol 1.0: Low-power, mid-range sensor and actuator communication.* In 2015 IEEE Conference on Standards for Communications and Networking (CSCN), pages 54–59, October 2015. doi:10.1109/CSCN.2015.7390420.

[3] R. Berkvens, B. Bellekens, and M. Weyn. *Signal strength indoor localization using a single DASH7 message.* In 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pages 1–7, Sapporo, September 2017. IEEE. Available from: http://ieeexplore.ieee.org/document/8115875/, doi:10.1109/IPIN.2017.8115875.

[4] *Murata partners with STMicro to add SigFox connectivity to its LoRaWAN module enabling "Best of Both Worlds" capability | Murata Manufacturing Co., Ltd.* Available from: https://www.murata.com/en-global/products/info/connectivitymodule/lpwa/2017/0718.

[5] F. Bendaoud, M. Abdennebi, and F. Didi. *Network Selection in Wireless Heterogeneous Networks: a Survey.* Journal of Telecommunications and Information Technology, 4:64–74, January 2019. Available from: https://www.itl.waw.pl/czasopisma/JTIT/2018/4/64.pdf, doi:10.26636/jtit.2018.126218.

[6] Wen-Tsuen Chen, Jen-Chu Liu, and Hsieh-Kuan Huang. *An adaptive scheme for vertical handoff in wireless overlay networks.* In Proceedings. Tenth International Conference on Parallel and Distributed Systems, 2004. ICPADS 2004., pages 541–548, July 2004. doi:10.1109/ICPADS.2004.1316136.

[7] J. Hoebeke, J. Haxhibeqiri, B. Moons, M. V. Eeghem, J. Rossey, A. Karagaac, and J. Famaey. *A Cloud-based Virtual Network Operator for Managing Multimodal LPWA Networks and Devices.* In 2018 3rd Cloudification of the Internet of Things (CIoT), pages 1–8, July 2018. doi:10.1109/CIOT.2018.8627134.

[8] J. Famaey, R. Berkvens, G. Ergeerts, E. D. Poorter, F. V. d. Abeele, T. Bolckmans, J. Hoebeke, and M. Weyn. *Flexible Multimodal Sub-*

Gigahertz Communication for Heterogeneous Internet of Things Applications. IEEE Communications Magazine, 56(7):146–153, July 2018. Available from: https://ieeexplore.ieee.org/document/8419195/, doi:10.1109/MCOM.2018.1700655.

[9] D. Castells-Rufas, A. Galin-Pons, and J. Carrabina. *The Regulation of Unlicensed Sub-GHz bands: Are Stronger Restrictions Required for LPWAN-based IoT Success?* page 17, November 2018. doi:arXiv:1812.00031.

[10] P. Wetterwalk, P. Thubert, and E. Levy-Abegnoli. *HETEROGENEOUS WIRELESS: DUAL LORA-NB-IOT REDUNDANT CONNECTIVITY*. page 6, 2018.

[11] M. Chen, Y. Miao, X. Jian, X. Wang, and I. Humar. *Cognitive-LPWAN: Towards Intelligent Wireless Services in Hybrid Low Power Wide Area Networks*. IEEE Transactions on Green Communications and Networking, pages 1–1, 2018. arXiv: 1810.00300. Available from: http://arxiv.org/abs/1810.00300, doi:10.1109/TGCN.2018.2873783.

[12] F. Lemic, A. Behboodi, J. Famaey, and R. Mathar. *Location-based Discovery and Vertical Handover in Heterogeneous Low-Power Wide-Area Networks*. IEEE Internet of Things Journal, pages 1–1, 2019. Available from: https://ieeexplore.ieee.org/document/8804217/, doi:10.1109/JIOT.2019.2935804.

[13] G. Castignani, A. E. Arcia Moret, and N. Montavont. *A study of the discovery process in 802.11 networks*. ACM SIGMOBILE Mobile Computing and Communications Review, 15(1):25–36, January 2011. Available from: https://hal.archives-ouvertes.fr/hal-00609309, doi:10.1145/1978622.1978626.

[14] M. Loy and R. Karingattil. *ISM-Band and Short Range Device Regulatory Compliance Overview*, 2005.

[15] A. Dunkels. *Design and Implementation of the lwIP TCP/IP Stack*, February 2001. Available from: https://www.artila.com/download/RIO/RIO-2010PG/lwip.pdf.

[16] Semtech. *sx1276-1278113.pdf*, 137. Available from: https://www.mouser.com/ds/2/761/sx1276-1278113.pdf.

5

Linked Data in Constrained Wireless Sensor Networks

In the previous chapters, a protocol stack was put forward in order to connect very constrained devices to the Internet and the World Wide Web (WWW). The technologies that were used make it possible to plug these devices into a Web of Things. However, not only do these devices require interoperability in order to access their resources, so does their data. As the future Web will evolve to a Semantic Web, where data can be interpreted by machines and humans, data coming from Low Power Wide Area Networks (LPWANs) should be semantically enriched. Therefore, this Chapter studies how these LPWANs can be connected using the technologies of the future Web.

⋆⋆⋆

Many applications and innovations arose from the fact that an excessive amount of information is openly available thanks to the open vision Tim Berners-Lee had about the WWW. With the rise of the Internet of Things (IoT), even more information will be available. As governments, individuals and companies are rolling out sensor networks to manage their daily tasks more easily, even more data is added to the ever expanding datasphere that is predicted to grow from 33 Zettabytes (ZB) in 2018 to 175 ZB by 2025 [1]. Since the real time collected data from the wide spectrum of physical sources that defines the IoT might be compared using time series analysis algorithms

to historically collected data, it will hold even more information in the near future. However, different data sources should be easy to integrate in order to unlock new scientific insights and to empower innovations. A fairly old concept for doing so, which gained popularity in academia to ensure free access to academic data, is the concept of Open Data. In order for this open data to be easily accessible and readable by machines, Tim Berners-Lee suggested the 5-star Linked Open Data concept in 2006, stating that open data should be linked to other data in a Linked Data Platform (LDP) to provide context [2]. Therefore, a set of operations are defined, which can be used to provide an architecture to read and write Linked Data (LD) on the web.

However, today two major problems exist. The first one being that data can most often only be retrieved from proprietary Application Programming Interfaces (APIs) and secondly, that these data are mainly stored by application providers. This led to innovations in the field of data decentralization, with Solid as the leader. Solid enables data reusability using open standards and protocols, following the 5-star Linked Open Data concept [3]. Nevertheless, as Solid currently mainly focuses on social web applications, where users can easily control where they store and who they share their personal data with. A similar approach should be maintained for easy integration of services and development of the Internet of Things data cloud. As the IoT world is also evolving, several open standards and data models are available. As these standards make it easier to integrate IoT devices with gateways or back-end platforms, perform device management and exchange data, they become increasingly adopted. However, as their design was driven by the device and network constraints often encountered in IoT environments, they do not incorporate linked open data features.

Therefore, as there is currently no straightforward solution present to store Internet of Things data in an open, decentralized manner, this paper analyses how data from constrained IoT networks can be incorporated in the Linked Open Data cloud. By doing so, a global Web of Things can be built in which any thing can be linked to any other thing. In order to do so, data should be self-describing, which requires the use of semantics. However, our evaluation shows that a naive implementation of these semantics using Linked Open Data concepts will increase the energy consumption of the sensor devices to a great extent, and thus requires a more suitable ontology at the sensor side. Furthermore, we show that by choosing the right protocol for such constrained networks, device management can be performed too, while decreasing the battery drain. This, however, requires a mapping from the wielded light-weight encoding schemes to any format supported by LDPs.

This chapter is based on the following publications:

- Bart Moons, Jeroen Hoebeke. *Towards an Open Web of Things.* Published in the proceedings of the 2020 IEEE International Symposium on Technology and Society (ISTAS), 12-15 November 2020. p. 176-179. Online.

- Bart Moons, Flor Sanders, Thijs Paelman, Jeroen Hoebeke. *Decentralized linked open data in constrained wireless sensor networks* Published in the proceedings of the 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS), 14 December 2020. Online.

5.1 Motivation

5.1.1 Open Standards

The main objective in developing IoT solutions and applications is to integrate everyday objects into our virtual world in order to gain more insight into our environment and manage our lives more efficiently. This integration, however, still leaves much room for improvement. Users often need to set parameters within the application, that must be reconfigured when the person's context changes [4]. In order for components (i.e. products and services) to work together and act upon context changes (e.g. turning the heating on when you start navigating home from work), interfaces that can understand each other are required. Ideally, these interfaces are something that we have collectively agreed upon in the ultimate goal of achieving a common language for information sharing between services, better known as open standards. The opposite of open standards are proprietary protocols and can be defined as a way of retrieving information, specific to one or more devices or services. Not only should the way to retrieve information be standardized, in order for systems to be able to seamlessly work together, data should be structured in order to be able to reuse data. If we want an application to be able to turn on the heating, when you start driving home, it should be able to interpret this data. The application should know how far your work is from your home and in what units you measure this distance. It should know what the temperature is in your house and in which units your sensor delivers this data. In order to achieve such interoperability, information models are required to represent a harmonised format that can be used by both data consumers and data publishers.

Multiple technologies exist to use information models and to structure data. The following subsections briefly discusses three of them.

5.1.1.1 RDF

The Resource Description Framework (RDF) is a framework for expressing information about resources and interchanging this information between machines without loss of meaning. An RDF statement is made up of a *subject*, *predicate* and *object* that can be used to represent the relationship between the subject and the object [5]. Only a single label is supported between subject and object, which can in some cases lead to complex statements. The RDF has been standardized by the World Wide Web Consortium (W3C), together with other semantic tools such as SPARQL Protocol and RDF Query Language (SPARQL) and Web Ontology Language (OWL). The standard defines a clear API, that all implementations must adhere to. In RDF there is already a very large amount of data available to link towards, such as DBpedia.

5.1.1.2 NGSI-LD

The Next Generation Service Interfaces-Linked Data (NGSI-LD) information model has been standardized by European Telecommunications Standards Institute (ETSI) following a request from the European Commission as part of the "Rolling Plan for ICT Standardisation" [6]. This plan has the ultimate goal of facilitating the open exchange of data between stakeholders of different domains. The NGSI-LD format is derived from Property Graphs (PGs). A PG is an attributed graph where the attributes of graph elements are represented as key-value pairs. In attributed graphs, nodes and edges can have an internal structure with the advantage of decreased storage size. However, the major drawback of having an internal structure is the risk of limiting expressiveness. For example, extending attributes with another set of attributes becomes difficult and consequently limits scalability [7]. Another drawback of the NGSI-LD information model is its architecture; a context broker is used to interconnect several components. This is less scalable, compared to a client-server architecture, where every component can evolve individually [8].

5.1.1.3 LPG

The Labeled Property Graph (LPG) is, just like RDF a graph data model that is used to describe, store and explore graph data. However, in contrast to the RDF model, nodes and edges in LPG can have an internal structure, just like the NGSI-LD information model, leading to a more compact data

representation [5]. However, most property graph solutions do not have a schema, resulting in use case specific mapping between solutions and also do not have a standard query language and exchange format.

In search of an interoperable solution, we can conclude from the above observations that the RDF is the most interoperable solution that fits the scope of this dissertation. Moreover, as the following Sections will demonstrate, current advancements in data decentralization are supported by RDF.

5.1.2 Responsible use of Data

The Web has evolved into an information space where we incessantly share data with each other. However, the information we share is largely controlled by large entities. Past scandals have shown how these entities can use the possession of our data to exercise economic and political control by sending targeted advertisements or possibly manipulate data (Cambridge Analytica, the Russian interference in the US presidential elections of 2016). Incidents like these show that the unregulated cyberspace can be misused to influence our society. In addition, the exponential growth of data during the last decade has caused a greater applicability of Artificial Intelligence (AI). AI algorithms are being used by scientists to make predictions and reconstructions, by banks trying to determine who is eligible for a loan and who is not, or by companies to screen applicants. AI systems can therefore contribute to finding patterns and help in making decisions, but also in assisting human biases. In short, responsible use of data is becoming more important every day.

5.1.3 SOLID

This is something that recently has been acknowledged by the inventor of the World Wide Web, Tim Berners-Lee, who took action by starting the Solid project in order to redistribute data and return it to its rightful owner [3]. Solid includes a series of standard formats for identity, comments and discussions, among others, to structure social data. The project also provides an ecosystem above which social applications can run to eliminate proprietary APIs. This is done by storing the user's data in a POD or Personal Online Datastore (POD) store that can be self-hosted or hosted by a POD provider. Users can distribute information among different pods; personal information, contact information, financial information, data about their health, etc. By distributing the information, benefits in terms of privacy and security arise because the data is no longer stored by one entity. Moreover, the use of a standard data format may stimulate innovation. For

example, several initiatives already arose to use a single format for health information in order to provide better scientific insight [9]. In addition, people are generating valuable data when tracking all sorts of activities. If ownership is handed to the user, software developers could offer them a platform to combine data generated from fitness activities and their health information. Furthermore, the use of a standard format would allow easier analysis of this data by, for example, (sports) doctors to prevent overload injuries. As a result, new applications are no longer limited by data that is scattered and inaccessible across different, closed platforms; users just have to agree to share them. In addition, the hardware and software market remain separate, since specialization in the development of embedded devices is not a requirement for software companies to access data and consequently data from any sensor device would be integrable in any platform.

In order to achieve this vision, Solid relies on the foundations of the LDP to make data self-descriptive and interpretable for machines. Semantically linked data gives scientists, among others, access to a database that spans multiple domains, which can significantly increase insight into major humanitarian problems, such as Alzheimer's and global warming. When the Web started linking documents with hyperlinks, the foundations of a global library had been laid. By connecting data using hyperlinks, the Linked Data platform can build a global database available to anyone.

5.1.4 Towards an (Open) Web of Things

Today, data from IoT devices, such as Google's Nest, are mainly stored and exposed in the same way as with many web applications. Data are collected by a central entity and proprietary APIs are (sporadically) offered for data retrieval. As a result, companies that offer IoT systems have valuable information about the living environment of users and will gain more insight into their lives. This can give more power to companies that own the data and consequently a greater imbalance can arise on the market. In order to prevent both economic and social imbalances, it is important to redistribute the data here as well.

In addition to consumers and businesses, governments will also obtain more and more valuable information by measuring our living environment. However, this data concerns information about public spaces, financed with public money. The Linked Open Data (LOD) movement already suggested in the past to make this data publicly available, assuming that this would increase the transparency and trust in the government. With the advent of the Internet of Things, this static open data might be supplemented with a tremendous amount of real-time data.

5.1.4.1 Linked Open Data

Linked Open Data has many advocates and opponents. An argument against LOD says that the large amount of publicly available information can actually lead to less understanding and more confusion. People can also make wrong assumptions, which would create less trust. It is therefore important to provide good infrastructure where people have a user-friendly platform which allows them to rely on collective intelligence. Another argument counters that decisions taken collectively are out of hands of policymakers, which consequently makes society less controllable. As open data will make society more open and driven by movements, this requires smart, easy-to-use infrastructure with feedback loops where the government can learn from its citizens and vice versa. Active monitoring of these feedback loops and conversations with people require an effort from local authorities, but at the same time gives citizens the feeling they can actively contribute. Counterparts also claim that the open data movement will further contribute to the digital divide, leveraging on what H.G. Wells said in 1951 that "statistical thinking one day will be as necessary for efficient citizenship as the ability to read and write" [10].

5.1.4.2 Open Cities

Although the arguments for publishing Linked Open Data are often based on a rather simplistic and idealized view and arguments against are based on the belief that barriers are so high that the opposite effect can be achieved, there is insufficient evidence to support a conclusive decision about the possible impact of Linked Open Data and the transition of governments to an open system. However, sensor data can play a major role in drawing conclusions from the beliefs surrounding the open publication of data. This can be done by employing the "comply or explain" principle to publish data as long as there is no argument against it. The relatively simple information provided by sensor data limits the required ability to have a deep understanding of the types of (causal) relationships and a profound knowledge of statistical techniques. A final argument in favor of open data argues that only a small team of developers from local governments has access to valuable data to initiate new applications where resources may lack. By openly publicizing sensor data, the opportunity to a whole community of creatives is offered to respond to local needs of neighborhoods by coming up with new, innovative solutions. For example, digital traffic signs indicating a temporary parking ban or parking sensors that actively monitor parking spots provide a wealthy source of information to local authorities. Much of this data, such as license plate number or even identity information, is strictly private and only relevant

for them. However, by openly publishing information such as the geolocation and the availability time of the parking spot, people and companies can add this to maps and applications, providing accurate real-time parking information. Open sensor data in cities have already been put forward in the past in order to stimulate innovation. Sidewalk Labs for example - the urban innovation organization of Alphabet Inc. - and Waterfront Toronto started in 2017 with the elaboration of the hypermodern Quayside project with the aim of making city life better for everyone [11]. By using open, digital infrastructure and clear standards for making the collected data about the environment publicly available, the team was hoping to create a positive cycle of urban innovation. Unfortunately, economic uncertainty forced the team to halt the project. Domingo et al. have also discussed the benefits of Open Sensor Data in cities and provided an architecture for doing so [12]. However, the centralized database in their proposed platform limits openness and scalability. A Linked Open Sensor Data platform must invest in offering feedback loops where discussions, proposals for decision-making or even private data can be added. By making use of a distributed system, such as Solid, to store and link the data, openness and scalability can be ensured. This may enable social machines such as citizen science, crowd sourcing and collective intelligence to continue to develop. This way of collecting and sharing data can also show how certain events influence our environment. Furthermore, a Linked (Open) Sensor Data platform offers a great source of information for researchers. Access to real-time data, which can be compared with historically collected data can provide a considerable insight into changes within our ecosystem. In order to realize this vision, two things are needed: the use of (1) standardized protocols for retrieving sensor data and (2) a distributed storage using open data formats. An architecture that complies with these requirements in the field of LPWAN is given in Section 5.4.

5.2 Related Work

Incorporating Internet of Things data to the Linked Open data is no novelty. The authors in [13] address the interoperability issues of the Internet of Everything, whereby proposing the idea of "Linking Everything". Their solution is based upon the 5-star Linked Open Data concept, replacing the Hyper Text Transfer Protocol (HTTP) by the Constrained Application Protocol (CoAP). Apart from their vision and enabling technologies, they mainly focus how real-time data can be queried from traditional data platforms such as Apache Jena, Virtuoso and Stardog. Similarly, in [14] a HTTP-CoAP proxy is implemented to translate CoAP messages to HTTP requests. Additionally,

new content-format types were introduced (i.e. `application/ld+json` and `text/turtle` to support LDP features. Both articles assume that devices can be queried directly without constrained links, being exactly a fundamental property of Low Power Wide Area Networks (LPWANs) frequently used for connecting constrained IoT devices. Other work considers the Java Data Objects (JDO) standard as a data model for a resource framework incorporated in a generic, uniform data management platform [15]. The authors present a homogeneous service layer as an interface to heterogeneous data sources. For this purpose, they defined a novel API which can be used for data validation and resources handling, thereby centralizing and unifying the data model from a broad range of data sources. Furthermore, a transparent proxy is provided, which can deliver data to any SPARQL endpoint. However, none of these solutions have incorporated data in a decentralized platform where the data producer can provide access control to the application of his choice.

5.3 Background Technologies

5.3.1 SOLID

As mentioned in the Introduction, Solid provides a platform for decentralized data storage which allows users to easily switch between application providers without having to reorganize their data. Solid does this by offering a platform where each user can store their data in an online storage space (called a pod), which can be either self hosted or provided by pod providers. Also developers can benefit from this approach by making use of data in a standardized format, instead of managing complex APIs offered by social Web applications.

In order to ensure data interoperability and easy integration with the current form of the Web, Solid relies as much as possible on existing open standards. Data is exchanged using HTTP and stored using the RDF. Semantic Web technologies, such as RDF, enable self-describing, linkable, yet independent, data management mechanisms. RDF is a standardized format for representing information on the Web. It describes a data model that serves to link RDF-based languages and specifications and as such provides the basis of the Linked Open Data cloud. RDF consists of graphs - formed out of self-descriptive subject-predicate-object triples - and datasets, used to organize collections of RDF graphs [16]. RDF data can be structured by using an ontology; a formal way of expressing a subject area and how their different properties are related. It is important for applications that their ontologies can be accessed by others so that data can be reused by groups

of potential consumers. Ontology sharing is the first step towards semantic interoperability on a larger scale. A next step includes ontology mapping and matching, where resources from different ontologies can be matched with others. A significant number of mapping tools exist to incorporate data into the Linked Data cloud. One of the few mapping tools, which allows to map heterogeneous sources into RDF, is the RDF Mapping Language (RML). In real-world situations, multiple sources of different formats can be part of a variety of data ecosystems. As RDF is often not supported by these data sources, the RML can be used as a generic mapping language to express custom mapping rules from heterogeneous data structures to the RDF data model.

Finally, different serialization syntaxes exist for storing and exchanging RDF such as Terse RDF Triple Language (Turtle) and JavaScript Object Notation (JSON)-LD (JSON-LD).

5.3.1.1 IoT ontologies

Multiple standards are currently being developed in order to provide semantic interoperability in the IoT. The Semantic Sensor Networks (SSN) Ontology is created by the Spatial Data on the Web Working Group. At its core lies the Sensor, Observation, Sample and Actuator (SOSA) ontology that is intended for light-weight and standalone use. The Smart Applications REFerence (SAREF) Ontology is another ontology that tries to achieve semantic interoperability in the domain of the IoT. In the scope of real-world deployments, such as in [17], both ontologies seem to have limited support for specialized use cases. However, SSN provides a higher level of abstraction (a System belonging to a Platform and a Deployment). While, SAREF only provides an abstraction that starts from the device itself. The deployment abstraction provided by the SSN/SOSA can be more interesting in the scope of LPWANs that typically encompass very large deployments.

5.3.2 LwM2M

As the Web became the largest connected information system through the use of standardized, resource oriented protocols, the Internet of Things should aim for a similar approach to become a true part of the Web ecosystem. Consequently, the Internet Engineering Task Force (IETF) developed the Constrained Application Protocol (CoAP). However, CoAP does not provide semantic interoperability; developers can specify themselves which Uniform Resource Identifiers (URIs) should be used to access resources. Therefore, on top of the CoAP protocol, the Light Weight Machine to Machine (LwM2M) specification from the Open Mobile Alliance (OMA) was developed in order

to provide a generic API to perform device bootstrapping, registration and management. Besides these interfaces, an object model is defined to represent collections of mandatory and optional resources that offer semantics to both device and management server and as such provide machine-readable representations of the semantics [18]. The provided object models are structured using pre-defined URI-templates, known to both client and server. An object can consist of several instances, which at their turn can consist of several resources. Objects, instances and resources are all identified with a unique identifier within its scope, making up the URI. These resources can then be accessed using CoAP operations.

5.4 Architecture

5.4.1 System Overview

In order to connect the constrained Internet of Things to the Linked Open Data cloud, several interfaces are required. First of all, the produced data will be transmitted over constrained wireless links to a receiving gateway. The data producers, depicted at the bottom of Figure 5.1, make use of LwM2M, a communication protocol that can fit every network and offering both device management and data transfer services.

The generated Internet Protocol (IP) packets will then be distributed to a LwM2M management server, where after the registration of a device a data flow can be established. Finally, packets of interest (i.e. packets containing sensor measurements) are filtered and forwarded to the Solid pod. As Solid makes use of a Web Access Control (WAC) mechanism for decentralized cross-domain authorization, multiple authorized applications and parties can be allowed to read, write, append and control the data. This makes it possible not only to have multiple clients writing data to the same pod, but also for applications to gather and combine data from different pods in a decentralized manner, with the possibility of opening up the data to the Linked Open Data Cloud [3].

5.4.2 Implementation

As described in the previous Section device management and data collection from several networks, such as NB-IoT, Long Range Wide Area Network (LoRaWAN) and IEEE 802.15.4, can be performed by means of a LwM2M server. For this purpose, Leshan, an open-source OMA LwM2M server implementation is set up (illustrated in the upper right corner of Figure 5.1). In order to pass every received message from Leshan to a supplementary data processor, Leshan exposes a RESTful (adhering to the Representational

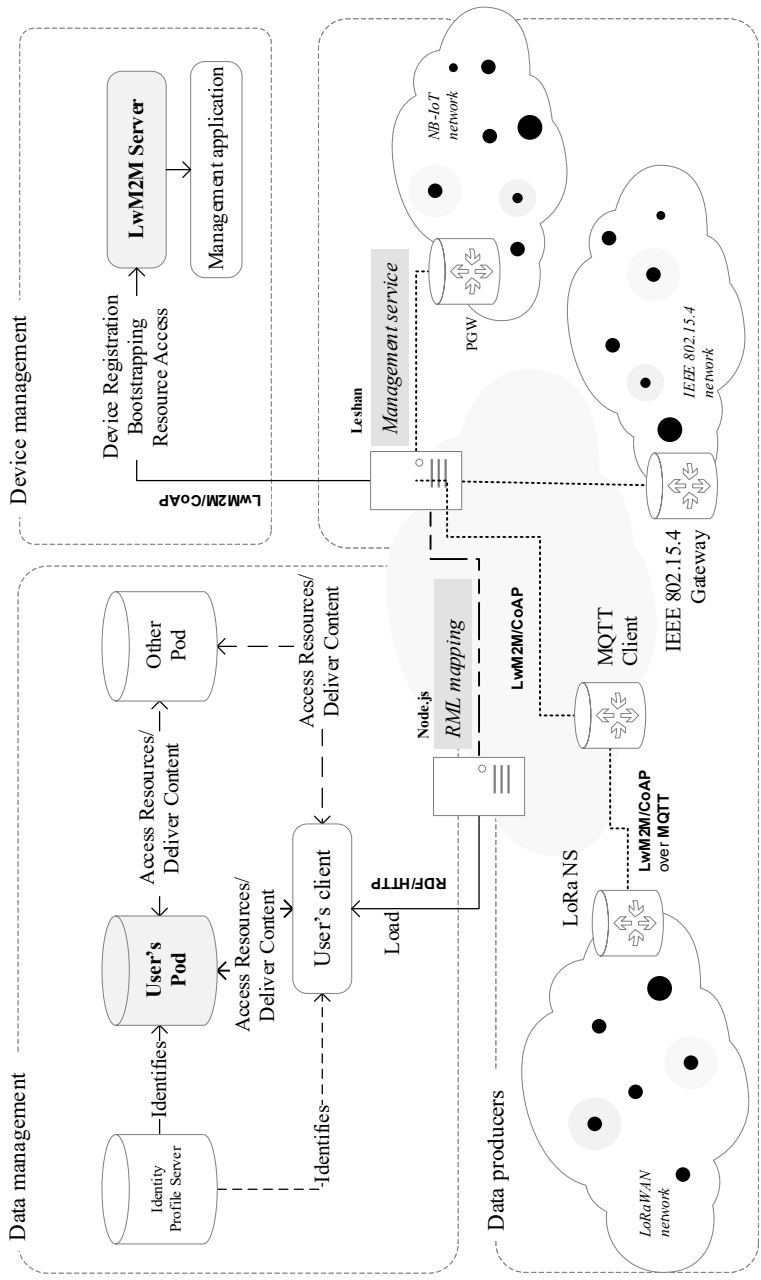


Figure 5.1: Implemented system overview. Data from LPWAN devices is captured and transmitted to the LwM2M server. A node.js instance subscribes to the Leshan event stream in order to map the LwM2M data to RDF and publish it to the Solid pod.

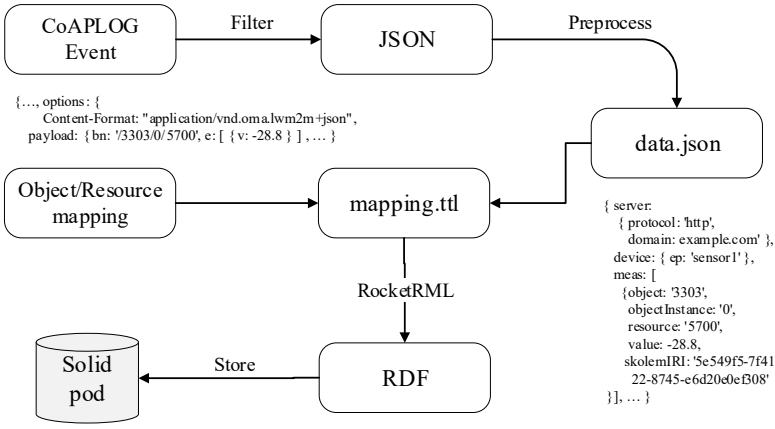


Figure 5.2: Internal working of the translation mechanism running on the node.js server.

State Transfer (REST) constraints) communication API between the front- and back-end. The LwM2M/CoAP messages of interest are forwarded to a Node.js application, where an RML script translates them to an RDF graph, required to store them in the Solid data pod. The RDF graph consist of nodes uniquely identified by a URI for which relations are defined using predicates. Every node in the graph represents either a device, a sensor instance or a single measurement. Once the graph with the updated data is produced by the RML script, the measurements are sent to the Solid pod. Existing graphs will merge fluently since every node with an existing URI is considered the same and therefore only new elements are stored. An overview of the translation flow is given in Figure 5.2.

As Section 5.3 introduced, the description of RDF data requires an ontology, which acts as a lexicon for the data model. As a result of a study for the European Commission a basis of an ontology containing the OMA LwM2M data structure is publicly available [19]. However, in order to store historical sensor data, various adaptations were required to this ontology [1]. Firstly, following the OMA LwM2M registry, the mandatory objects and their resources were completed and predicates were added to represent the relations between them. Next, a device object was added to represent the OMA LwM2M client and its exposed objects. Finally, the `hasTimestamp` predicate was added to the `SensorValue` resource enabling the storage of

[1] https://iotsolidugent.inrupt.net/public/ontologies/omalwm2m.owl.ttl

historical data in a simple manner.

5.5 Evaluation

One of the main requirements for low power sensors is to operate on a single battery charge for multiple years and therefore consume as little energy as possible. Since wireless communication modules tend to have the highest energy consumption in IoT sensor devices, the communication overhead should be kept as small as possible. This Section will evaluate the aforementioned encodings and ontologies in order to select the most appropriate solution to initiate IoT sensor devices into the semantic web of things. Previous work around this topic introduced additional content-format types in order to incorporate these sensors directly in the Web as Linked Data providers [14]. The authors released their implementation as open source [2], which was used during the comparison of the different ontologies and encodings.

In order to decide which ontology suits constrained networks the best, the overhead of a CoAP `GET` request to the memory availability of the device and its corresponding response is evaluated for the Linked Data content-format types `application/ld+json`, `text/turtle` and the LwM2M `text/plain`, `application/vnd.oma.lwm2m+json` and `application/vnd.oma.lwm2m+tlv` data types.

5.5.1 CoAP CON request

Due to the request-response nature of CoAP, every response is preceded by a request from a client. First, the linked data request is evaluated and can been seen in Listing 5.1.

Listing 5.1: CoAP JSON-LD GET request

```
CON, MID:38929, GET, TKN:61 62 63 64 65 66 67 68, /
    sensors/memory
```

On the contrary, the request in Listing 5.2 shows how the same resource can be queried using the predefined LwM2M URI-templates.

Listing 5.2: CoAP LwM2M GET request

```
CON, MID:35534, GET, TKN:ee 10 01 6d 28 3b 79 6d,
    /3/0/21
```

[2] https://github.com/sisinflab-swot/ldp-coap-framework

It can be seen from Figure 5.3 that LwM2M provides a higher efficiency in CoAP `CON` requests due to the object model known to both server and device. Even with User Datagram Protocol (UDP) and Internet Protocol Version 4 (IPv4) overhead, a LwM2M client will not surpass the fragmentation limit of Long Range (LoRa) using Spreading Factor (SF) 12, which would require multiple packets and consequently consume a lot more energy.

5.5.2 CoAP response

In response to the `GET` request, the server will reply with the value of the queried resource. Listing 5.3 shows the payload of a response in the `application/ld+json` data format.

Listing 5.3: Payload of a JSON-LD response to a CoAP GET request

```
{
  "@id": "coap://192.168.2.16/
     MEM_IBCN_096_202006160959AM",
  "@type": "http://www.w3.org/ns/sosa#Observation",
  "http://www.w3.org/ns/sosa#hasSimpleResult": {
    "@type": "http://w3id.org/lindt/
       custom_datatypes#ucum",
    "@value": "253952 kB"
  },
  "http://www.w3.org/ns/sosa#madeBySensor": {
    "@id": "coap://192.168.2.16/M-2-2230"
  },
  "http://www.w3.org/ns/sosa#observedProperty": {
    "@id": "coap://192.168.2.16/Memory"
  },
  "http://www.w3.org/ns/sosa#resultTime": {
    "@type": "http://www.w3.org/2001/XMLSchema#
       dateTime",
    "@value": "2020-06-16T09:59:14.141+02:00"
  }
}
```

Listing 5.4 clearly shows that this is in sheer contrast with the payload size of a LwM2M request encoded as `application/vnd.oma.lwm2m+json`.

Listing 5.4: Payload of a LwM2M JSON response containing multiple historical representations of a memory resource

```
{
  "bn":"/3/0/21", "bt":1592258400,
  "e":[
```

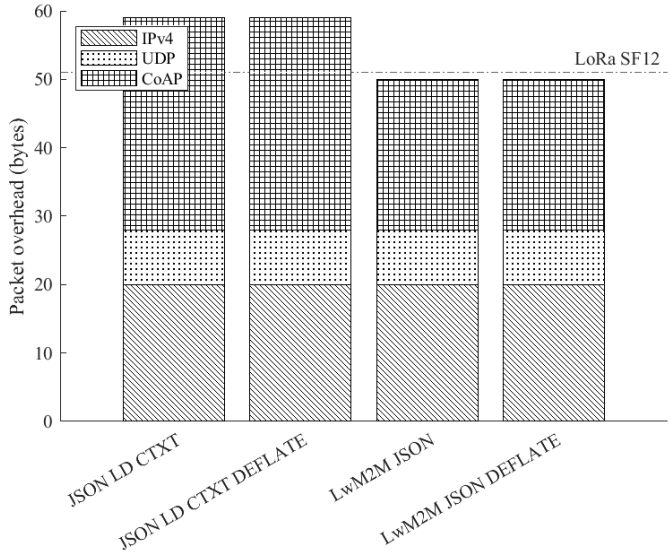


Figure 5.3: GET request performed from client to server with different ontologies and encodings

```
    {"n":"","v":253901, "t":-5},
    {"n":"","v":253952, "t":-35}
  ]
}
```

5.5.2.1 JSON-LD optimization

However, in JSON-LD, it is possible to provide a context in order to use simple, local names, while maintaining semantic meaning beyond the initial exchange partners. An example context for the naive request from Listing 5.3 is shown in Listing 5.5. The optimized response is given in Listing 5.6. This Listing also shows how we optimized the timestamp with the owl-time ontology and the observation id as a simple counter. Furthermore, in order to make a fair comparison with LwM2M, we removed the `madeBySensor` property. However, in order to show some of the LwM2M optimizations, we introduced a second historical measurement.

Listing 5.5: The context of the JSON-LD request from Listing 5.6

```
{
  "@context": {
    "@base": "coap://192.168.2.16",
```

```
    "sosa": "http://www.w3.org/ns/sosa#",
    "cdt": "http://w3id.org/lindt/custom_datatypes
       #",
    "owlt": "http://www.w3.org/2006/time#",
    "v": "@value",
    "n": "@id",
    "u": "@type",
    "sr": "sosa:hasSimpleResult",
    "sp": "sosa:observedProperty",
    "st": "sosa:resultTime",
    "so": "sosa:Observation",
    "ot": "owlt:numericPosition",
    "cu": "cdt:ucum"
  }
}
```

Listing 5.6: The payload of a JSON-LD request can be compressed using a context

```
[
  {
    "n": "1", "u": "so",
    "sr": {"u": "cu", "v": "253952 kB"},
    "sp": {"n": "Memory"},
    "st": {"u": "xt", "v": "1592258395"}
  },
  {
    "n": "2", "u": "so",
    "sr": {"u": "cu", "v": "253901 kB"},
    "sp": {"n": "Memory"},
    "st": {"u": "xt", "v": "1592258365"}
  }
]
```

Figure 5.4 illustrates how the different encodings and ontologies relate to each other in response to a `GET` request. The fragmentation lines of different technologies are shown on the right and again indicate how LwM2M has a lower overhead. However, both encoding schemes introduce fragmentation over a LoRa SF12 connection.

Even though, the responses can be optimized using the JSON-LD `@context` property, it can be seen from both figures that low power sensor devices can still benefit from the wielded approach where low power IoT ontologies are mapped using RML instead of using semantic web ontologies directly on top of IoT protocols.

Finally, in Figure 5.5 the battery lifetime of a LoRa SF12 device that

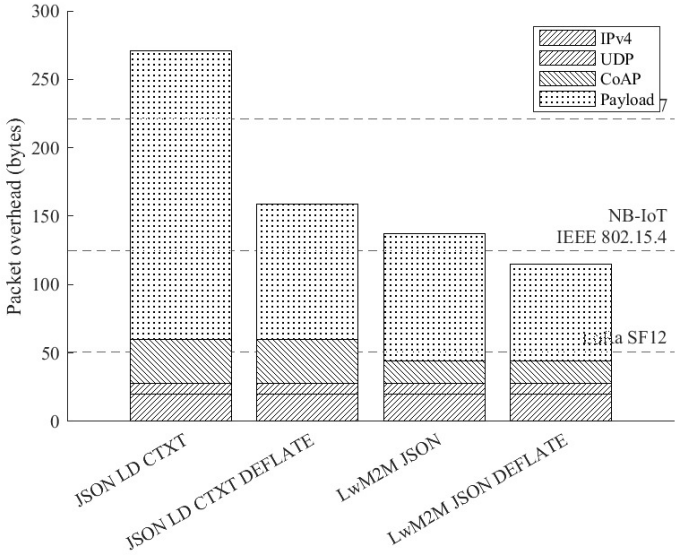


Figure 5.4: Response to a GET request, containing two historical measurements, performed from server to client with different ontologies and encodings

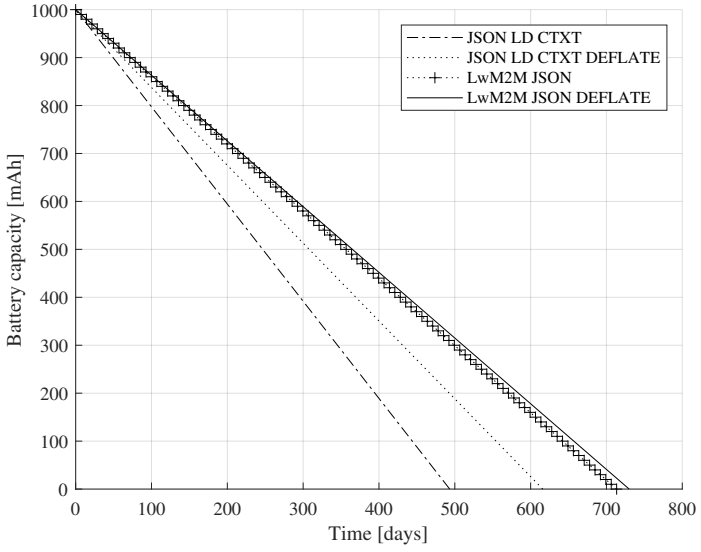


Figure 5.5: Battery drain of a LoRa SF12 communicating a single measurement with the server every hour, without self-discharge

responds every hour with a single new measurement to an initial `OBSERVE` request in different encoding schemes is shown. This clearly indicates how a light-weight encoding schema can decrease the energy consumption.

5.6 Conclusion

As current data storage lacks in true data ownership and data openness, novel principles are being developed, with Solid as the leader. This paper showed that it is possible to use a standard based approach over constrained wireless links to enable the concepts of Linked (Open) Data in the IoT. We showed that using our implementation less packets, and consequently less energy, is required for devices that should be capable to operate on a single battery charge for years, compared to a naive RDF-based approach. By using RML as an ontology matching tool, we showed that it is possible to couple a set of constrained IoT devices using the LwM2M management protocol to an RDF-based Solid POD.

References

[1] D. Reinsel, J. Gantz, and J. Rydning. *The Digitization of the World from Edge to Core*. page 28, 2018.

[2] *5-star Open Data*. Available from: http://5stardata.info/en/.

[3] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, and T. Berners-Lee. *Solid: A Platform for Decentralized Social Applications Based on Linked Data*. page 16.

[4] J. Miranda, N. Makitalo, J. Garcia-Alonso, J. Berrocal, T. Mikkonen, C. Canal, and J. M. Murillo. *From the Internet of Things to the Internet of People*. IEEE Internet Computing, 19(2):40–47, March 2015. Available from: http://ieeexplore.ieee.org/document/7061811/, doi:10.1109/MIC.2015.24.

[5] S. Purohit, N. Van, and G. Chin. *Semantic Property Graph for Scalable Knowledge Graph Analytics*. arXiv:2009.07410 [cs], September 2020. arXiv: 2009.07410. Available from: http://arxiv.org/abs/2009.07410.

[6] *Rolling Plan 2021 | Joinup*. Available from: https://joinup.ec.europa.eu/collection/rolling-plan-ict-standardisation/rolling-plan-2021.

[7] M. Margitus, G. Tauer, and M. Sudit. *RDF versus attributed graphs: The war for the best graph representation*. In 2015 18th International Conference on Information Fusion (Fusion), pages 200–206, July 2015.

[8] G. Privat. *ETSI 06921 Sophia Antipolis CEDEX, France Tel +33 4 92 94 42 00 info@etsi.org www.etsi.org*. page 55.

[9] Sciensano. *healthdata.be*, August 2020. Available from: https://healthdata.sciensano.be/en.

[10] M. Janssen, Y. Charalabidis, and A. Zuiderwijk. *Benefits, Adoption Barriers and Myths of Open Data and Open Government*. Information Systems Management, 29(4):258–268, September 2012. Available from: http://www.tandfonline.com/doi/abs/10.1080/10580530.2012.716740, doi:10.1080/10580530.2012.716740.

[11] S. Labs. *Master Innovation and Development plan, "Toronto Tomorrow: A New Approach for Inclusive Growth."*, June 2019.

[12] A. Domingo, B. Bellalta, M. Palacin, M. Oliver, and E. Almirall. *Public Open Sensor Data: Revolutionizing Smart Cities*. IEEE Technology and Society Magazine, 32(4):50–56, 2013. Available from: http://ieeexplore.ieee.org/document/6679327/, doi:10.1109/MTS.2013.2286421.

[13] D. Le-Phuoc and M. Hauswirth. *Linked Data for Internet of Everything.* In R. Gravina, C. E. Palau, M. Manso, A. Liotta, and G. Fortino, editors, Integration, Interconnection, and Interoperability of IoT Systems, pages 129–148. Springer International Publishing, Cham, 2018. Series Title: Internet of Things. Available from: http://link.springer.com/10.1007/978-3-319-61300-0_7, doi:10.1007/978-3-319-61300-0_7.

[14] G. Loseto, S. Ieva, F. Gramegna, M. Ruta, F. Scioscia, and E. Di Sciascio. *Linking the Web of Things: LDP-CoAP Mapping.* Procedia Computer Science, 83:1182–1187, 2016. Available from: https://linkinghub.elsevier.com/retrieve/pii/S1877050916302770, doi:10.1016/j.procs.2016.04.244.

[15] J. Pullmann and Y. Mohamad. *Linked Data Services for Internet of Things.* In Proceedings of the 2015 International Conference on Recent Advances in Computer Systems, Hail, Saudi Arabia, 2016. Atlantis Press. Available from: http://www.atlantis-press.com/php/paper-details.php?id=25847799, doi:10.2991/racs-15.2016.26.

[16] *RDF 1.1 Concepts and Abstract Syntax.* Available from: https://www.w3.org/TR/rdf11-concepts/.

[17] M. Poveda-Villalon, Q.-D. Nguyen, C. Roussey, C. De Vaulx, and J.-P. Chanet. *Ontological requirement specification for smart irrigation systems: a SOSA/SSN and SAREF comparison.* In 9th International Semantic Sensor Networks Workshop (SSN 2018), volume 2213, page 16, Monterey, United States, October 2018. CEUR Workshop Proceedings. Available from: https://hal.archives-ouvertes.fr/hal-02042584.

[18] O. LwM2M. *Lightweight Machine to Machine Technical Specification.* Available from: http://www.openmobilealliance.org/release/LightweightM2M/V1_1-20180710-A/OMA-TS-LightweightM2M_Core-V1_1-20180710-A.pdf.

[19] L. Daniele, F. den Hartog, and J. Roes. *Study on Semantic Assets for Smart Appliances Interoperability : D-S4: FINAL REPORT.* 2015.

6

FLINT: Flows for the Internet of Things

Chapter 4 presented an intermediary to provide a homogeneous interface towards the Static Context Header Compression (SCHC) router and to keep track of the last active technology. In Chapter 5, an intermediary was required to enrich Light Weight Machine to Machine (LwM2M) data with semantics. This Chapter leverages on these observations in the scope of the Port of the Future. As a result, a generic architecture is presented that is able to provide interoperability on five levels for Low Power Wide Area Network (LPWAN) devices.

⋆⋆⋆

This chapter is based on the homonymous article by

B. Moons, M. Aernouts, V. Bracke, B. Volckaert and J. Hoebeke

Published in MDPI Applied Sciences 11(19), 7th October 2021

Abstract New protocols and technologies are continuously competing in the Internet of Things. This has resulted in a fragmented landscape that complicates the integration of different solutions. Standardization efforts try to avoid this problem, however within a certain ecosystem, multiple standards still require integration to enable trans-sector innovation. Moreover, existing devices require transformations to fit in an ecosystem. In this paper, we discuss several integration problems in the field of Low Power Wide Area Networks in the context of the Port of the Future and propose a new distributed platform architecture, called FLINT. FLINT is a framework to program flexible and configurable flows on a per device basis. A flow is constructed from fine-grained components, called adapters. Due to the modularity of an adapter, users can easily integrate existing software. We evaluated FLINT based on five levels of interoperability and show that FLINT can be used to interconnect non-interoperable systems and protocols on every level. We have also implemented FLINT in a container based environment and demonstrated that a basic configuration has a 99% forwarding rate of 17.500 513-byte packets per second, showing that the architecture can deliver good performance.

6.1 Introduction

Due to the plethora of Internet of Things (IoT) technologies and standards available, an overarching platform is required to interconnect heterogeneous sources of data. The purpose of an IoT platform is to match non-interoperable data sources, ranging from something as small as a device to something as large as a platform spanning multiple application domains. This requires extensive flexibility. Unfortunately, many platforms, such as Amazon Web Service (AWS) IoT [1], Kaa IoT [2] and ThingSpeak [3], have closed, inflexible designs and do not enable low-level interaction between components in the system. Furthermore, extending an IoT system with existing applications and software libraries can often be complex due to the imposed programming paradigms and/or languages. Finally, the upcoming trend of mobile IoT devices that can switch between networks on the one hand and very constrained devices on the other hand, requires a platform aware of the device its connection state and the network it is connected to.

To address this, we present FLINT: a flexible, modular and scalable network architecture for interconnecting IoT devices, networks, middleware and platforms. A FLINT configuration can be built from fine-grained components on a per device basis. These components are data processing elements called *adapters*. An adapter consists of two sub-elements: an *agent* and a *sink*. The sink connects the adapter to the platform by means of a

Message Broker. The Message Broker provides a message bus for efficient data delivery using the publish/subscribe paradigm. The sink and the agent communicate over a socket interface so that the agent can be built from any programming language. This allows users to recycle their existing programs and implementations. A FLINT configuration can be built by chaining adapters. The user chooses a collection of adapters for a given device and connects them into a chain that represents the path the data will follow.

We have implemented FLINT in a container based environment and deployed it in Kubernets cluster to evaluate its scalability. A basic FLINT configuration has a 99% forwarding rate of 17.500 513-byte packets per second on a 2.4 GHz intel E5645. The 1% packet loss is likely caused by the Message Queuing Telemetry Transport (MQTT) broker dropping packets at such high throughput. Our evaluation shows that, compared to other platforms, FLINT's architecture is still able to deliver good performance. We show that the forwarding rate drops significantly when adding more hops to the system. However, by demonstrating that the modular architecture can distribute the load over different machines we show that the platform is able to scale for more demanding applications. Next, we evaluated FLINT based on five levels of interoperability: syntactic interoperability, device interoperability, network interoperability, semantic interoperability and platform interoperability. We thereby show that FLINT provides tools to cover all levels of interoperability as described in [4]. This is illustrated by our experiments in a Port of the Future context (Section 6.2), though can be extrapolated to any other use case requiring IoT connectivity integration. As a final contribution, FLINT is available as open source under the LGPL-3.0 license.

The remainder of this paper first analyzes other IoT platforms and their focus (Section 6.3) before describing FLINT's architecture (Section 6.4) which is evaluated in Section 6.5. Finally, Section 6.6 present conclusions.

6.2 Case Study—Port of the Future

The Port of the Future will be equipped with technology to cope with day-to-day challenges more efficiently [5]. Sensor measurements can be used to analyze and monitor ports in order to make better operational decisions. For example, transportation can be tracked between different port terminals to increase the coordination of traffic and better manage transport logistic chains. Furthermore, the environment, engineering structures, vehicles and vessels can be tracked in order to optimize their behaviour and contribute to sustainable ports. This either involves (1) existing data sources which have already adopted a specific data format or (2) (wireless) IoT devices.

In order to benefit from information that is already available (1), data

from existing systems, legacy systems and systems that support different standards, should be converted to a common format [6].

Furthermore, depending on the available wireless infrastructure of the port, this data will originate from a multitude of data sources (2):

- Multiple wireless communication networks can coexist that serve devices equipped with a single radio;
- Devices can be equipped with multiple radios that connect to different networks over time;
- Devices can have a radio that supports multiple modulation schemes.

For example, devices equipped with a sub-Gigahertz (GHz) compatible radio and multiple modulation schemes, such as Long Range (LoRa), DASH-7 (Gaussian Frequency Shift Keying (GFSK)) and Sigfox (Ultra Narrowband (UNB)), can alternate between long range, low throughput and medium range, higher throughput technologies [7]. FLINT was initially developed to interconnect such networks and devices. However, multiple cases, such as the integration with Obelisk [8], made clear that its flexible design was also well suited for fast prototyping and use case support. Therefore, this section presents several modules that were developed and illustrates the evolution of FLINT from a simple tool to a scalable IoT platform.

6.2.1 Heterogeneous LPWAN

The typical size of a port requires wireless technologies that can transmit traffic over large distances. Low Power Wide Area Networks (LPWANs) can span very large areas, however are characterized by severe bandwidth constraints [9]. Traffic towards the wireless network is often restricted by duty cycle limitations and must be operated using different transmission schemes. Furthermore, incorporating multiple wireless networks into a single platform requires some sort of routing. This can be achieved with a central entity that stores the last active network and the properties of all available networks.

Due to their bandwidth constraints, LPWANs often use proprietary payload encoding formats. This limits the application portability and interoperability among systems. In order to solve these issues, the Internet Engineering Task Force (IETF) LPWAN Working Group (WG) developed the Static Context Header Compression (SCHC) technique. This standard uses a *static* context to represent the most common Internet Protocol Version 6 (IPv6), User Datagram Protocol (UDP) and Constrained Application Protocol (CoAP) patterns of the sensor node, known to both the end-device

and the gateway. Every flow is distinguished by means of a unique identifier that precedes the payload of the message. This identifier is used by the translating gateway to perform the decompression of the message [10].

This way, very constrained devices can establish an end-to-end IPv6 link over a low bandwidth technology. However, LPWAN systems typically consist of a star topology in which an intermediary captures packets from receiving gateways to remove duplicates. The SCHC router [11] (i.e., the edge router that (de)compresses packets from the IPv6 network) must therefore implement an abstraction to capture packets from the intermediary. Typical examples of such abstractions include the MQTT, the Hyper Text Transfer Protocol (HTTP) and the Advanced Message Queuing Protocol (AMQP). These observations indicate the need to incorporate abstractions for different wireless networks and devices that have multiple network interfaces and their mapping to standardized outputs.

6.2.2 Localization

Location-based Services (LBSs) are an essential aspect of IoT applications, as they provide valuable context information. For instance, sensor measurements from IoT devices in a port can only be interpreted properly if they are correlated with the correct measurement location. Typically, Global Navigation Satellite Systems (GNSSs) such as Global Positioning System (GPS), Global Navigation Satellite System (GLONASS), BeiDou Navigation Satellite System (BDS) and Galileo are used for outdoor localization. Cellular networks or LPWANs are used to transmit location data from GNSS receivers on IoT devices to an IoT platform. To reduce the overall cost of IoT devices or to enable outdoor localization on devices with an extremely low energy budget, it is possible to omit GNSS receivers and apply localization methods such as Time Difference Of Arrival (TDoA), Angle of Arrival (AoA) or Received Signal Strength (RSS)-based methods to the LPWANs instead [12–14]. These methods are especially popular for indoor localization scenarios. Technologies such as Wi-Fi, Bluetooth Low Energy (BLE) or Ultra Wideband (UWB), can cope with the fact that GNSS coverage is not available in indoor environments [15].

Hence, an LBS can obtain location information from a wide range of data sources, depending on the current environment and active technology of an IoT device. Since the performance of a wireless localization method strongly depends on estimation errors related to the environment, the network deployment and the IoT device itself [16], LBSs must be able to intelligently switch between data sources while estimating their reliability [17]. Therefore, there is a need for flexible, heterogeneous platforms such as FLINT that can incorporate an additional module for localization purposes. Specifically

for a port use case, this module can switch between outdoor localization with GNSS if a valid fix is available, TDoA localization with Long Range Wide Area Network (LoRaWAN) if a transmission is received by at least four gateways, or accurate indoor fingerprinting localization when FLINT detects an active Wi-Fi network.

6.2.3 Data Transformation

In the past, ports, port services and general services already have developed systems to improve their day-to-day operations. By interconnecting (existing) infrastructure, the efficiency of the entire port can improve even more. Connecting non-interoperable platforms is often addressed as *platform interoperability* and requires data transformation. In order to do so, traffic must be transformed and interconnected between several platforms.

A common way of interconnecting data is by means of the Linked (Open) Data schema [1]. Linked Data (LD) can be the result of applying the collection of Semantic Web technologies, such as (Resource Description Framework (RDF), Web Ontology Language (OWL), etc.) on non-interoperable data. These tools are standardized by the World Wide Web Consortium (W3C) to provide an environment where applications can query (using SPARQL) and interconnect data from several domains [18]. Converting real world data to interlinked data can be challenging, due to their heterogeneity. This can be done generically using the RDF Mapping Language (RML) [19]. RML uses a Turtle mapfile that defines customized mapping rules, that can be applied to heterogeneous data sources. However, managing multiple data sources requires a specific mapfile on a per device basis.

Another common problem exists in the management of wireless equipment and IoT devices. In order to limit coding and integration effort it can be useful to have a unified mechanism to manage such wireless tools. For this, several management protocols are available today [20]. However, integration of already deployed infrastructure and sensors with standardized solutions can be hard. In this paper, we focus on the Open Mobile Alliance (OMA) LwM2M protocol [21]. FLINT tries to minimize the required effort so that their proprietary format can be transformed into LwM2M compliant equipment.

An abstract overview of the required components is given in Figure 6.1.

[1] https://5stardata.info/en/) (accessed on 14/07/2021)

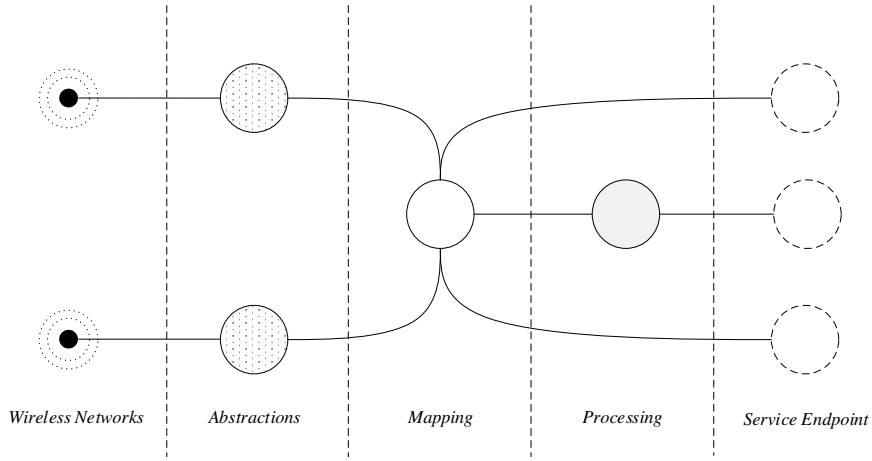


Figure 6.1: An abstract representation of the system. Wireless sensor networks on the left are integrated using Application Programming Interfaces (APIs). Their data are collected in a central point that provides an abstraction for devices with multiple network interfaces. From this central point, data are distributed to processing nodes and other platforms.

6.3 Related Work

A plethora of both open-source and commercial IoT platforms already exist. An overview of the most common systems is given in Table 6.1. The architecture of every platform is presented using a simplified notation representing the number of inputs, the number of intermediary processing elements and the number of outputs:

- N* - N* - 1: there can be N inputs, N processing elements and only 1 output. This architecture often stores the data inside the platform.
- N* - N* - N*/N*N*/X*: there can be N inputs, N processing elements and N, $N * N$, or X outputs.

With

- **N*** representing a custom, reusable element, either input, processing or output that can be implemented by the user.
- **X*** representing an element, either input, processing or output offered as is by the platform.

Some platforms allow communication between elements inside the deployed system. This is visualized using N*N*. Flows that end with a single

N* or X* provide elements that have a single output. Contrarily, flows that employ an N*N* approach provide reusable components for every node that can be integrated in the platform, i.e., every element has multiple inputs and outputs. An N*N* approach is illustrated in Figure 6.1, where the message can be passed back and forth between Abstractions, Processing elements and elements that provide platform interoperability at the Service Endpoint.

Next, the supported directionality of a platform is indicated using the Input/Output (I/O) column. Furthermore, the complexity of the platform is presented in column 'Complexity'. A distinction is made between low, medium and high complexity. A low complexity platform indicates that flows can be created by making use of a GUI. A platform that requires some technological background about the platform and technologies is labeled with medium complexity. A platform that requires in depth knowledge about the platform and the underlying technologies is labeled with high complexity.

Many of these projects aim to provide a user friendly interface to interconnect systems and applications. The open source flow-based editor, *Node-RED*, provides a Graphical User Interface (GUI) where modules can be connected to develop a flow [22]. Extra modules are also shared by the *Node-RED* community. *Node-RED* runs as a single Node.js instance and focuses on usability and fast prototyping, rather than on scalability and interoperability.

Another open source project, *node-wot*, implements the Web of Things (WoT) specification and aims to provide an abstraction between physical things and the current World Wide Web (WWW) [23]. This abstraction makes it possible to access things using web technologies. The W3C Thing Descriptions (TDs) provide semantic interoperability. Devices can be accessed and interconnected using LD concepts. *node-wot* runs inside a Node.js server and focuses on interoperability and usability, rather than on scalability.

INTER-IoT grew out of the objective to design and implement a cross-layer framework to provide interoperability among heterogeneous IoT platforms. [24]. The INTER-IoT Framework provides an API that exposes interactions to any IoT platform. Developers can then access the underlying IoT platforms through a single interface, thereby mainly focusing on interoperability.

Finally, *Eclipse Hono* provides a container based IoT platform. Different networks can be added by means of protocol adapters. Setting up an instance of *Eclipse Hono* can be done rather quickly using Kubernetes deployment tools. Devices can be added using an HTTP interface.

Project	Solution	License	Protocols	Usability	Interoperability	Architecture	Language	Direction	Complexity
AWS IoT [1]	Message broker, digital twin, Alexa Voice Service (AVS)	Commercial license, open source libraries	MQTT, HTTP	Draw flows in a GUI	No	N*-N*-X*	GUI	I/O	Low
OpenRemote [25]	Integrate assets, data visualization	AGPLv3	HTTP, Web Sockets (WS), MQTT, custom	Draw flows in a GUI	No	N*-N*-1	GUI	I	Low
Kaa [2]	Flows	Commercial license, free plan up to 5 devices	MQTT, HTTP	Draw flows in a GUI	No	N*-N*-1	GUI	I/O	Low
ThingSpeak [3]	Data visualization	Commercial license	WS, MQTT, HTTP	Integrate things in Matlab	No	N*-1-1	Matlab	I/O	Low

Project	Solution	License	Protocols	Usability	Inter-operability	Architecture	Language	Direction	Complexity
INTER-IoT [26]	Multi-layered approach	Apache-2.0	HTTP, CoAP, MQTT	Distributed platform	Yes	N*-N*-N*	Java	I/O	High
ThingsBoard [27]	Flows	Apache-2.0/Commercial	HTTP, MQTT, UDP, TCP	Draw flows in a GUI	No	N*-N*-X	GUI	I/O	Low
Eclipse node-wot [23]	Web of Things platform	EPL-2.0	HTTP, CoAP, WS, MQTT, custom	Access things using web technologies	Yes	N*-N*-1	Java Script	I/O	High
Eclipse Hono [28]	Container based IoT-platform	EPL-2.0	HTTP, MQTT, custom	HTTP endpoint	No	N*-N*-N*	Any [2]	I/O	High
Node-RED [22]	Visually integrate data flows	Apache-2.0	HTTP, MQTT, Web-sockets, custom	Draw flows in a GUI	No	N*-N*-N*N*	Java Script	I/O	Low

[2] Any programming language that supports AMQP

Project	Solution	License	Protocols	Usability	Inter-operability	Architecture	Language	Direction	Complexity
FLINT	Flows	LGPL-3.0	HTTP, MQTT, custom	Program flows	Yes	N*-N*-N*N*	Any [3]	I/O	Medium

Table 6.1: Non-exhaustive list of commercial and open source IoT platforms.

[3] Any programming language that supports sockets

FLINT tries to combine these properties and proposes an open source IoT platform with support for devices with multiple network interfaces, intended for fast prototyping, while maintaining scalability. FLINT can be used to interconnect platforms, while most platforms keep data locked in the platform or do not directly support platform interoperability.

6.4 Flint Architecture

A FLINT configuration can be built from fine-grained components on a per device basis. These components are data processing elements called *adapters*. An *adapter* represents a unit of processing. This can be seen as the transformation between two data sources to match the format from heterogeneous data sources, to enrich data by performing complex computations, to add semantics, etc. A FLINT configuration is a graph with *agents* at the vertices. An edge, or *sink*, between two agents represents a path for data transformation and/or data output. The user determines the configuration for a device or a fleet of devices by choosing a set of adapters between them. A running system consists of one or more Kubernetes deployments, made up of at least two containers that contain the sink and the agent. Several *sinks* can communicate via a Message Broker (currently MQTT), that provides a (distributed) message bus. The most important properties of an adapter are the following:

- *Adapter identifier.* Each adapter has a unique identifier. This specifies the topic that is used by the Message Broker to interface between adapters.
- *Configuration files.* An agent and a sink have a configuration file. These files are passed to the containers at initialization time. Adapters use these configuration files to set a per-adapter state and provide flexibility during deployment.
- *Socket connection.* An agent and a sink interface over a socket connection. Incoming data from the platform are forwarded to the agent. Modified data or data coming from other data sources is returned to the sink over the socket connection.
- *Sink port.* The sink port is an interface from the sink to the Message Broker. Data flows from the output sink port of an adapter to the input sink port of another adapter.
- *Agent port.* The agent port provides an optional connection to implementation specific interfaces. An input agent port, for example, can be

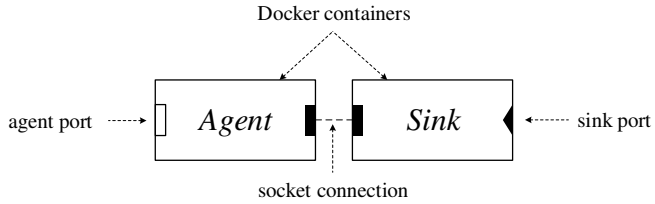


Figure 6.2: The basic components of an adapter. Triangular ports connect to the platform, filled rectangular ports connect the agent and the sink. Open rectangular ports connect the agent to non-FLINT sources.

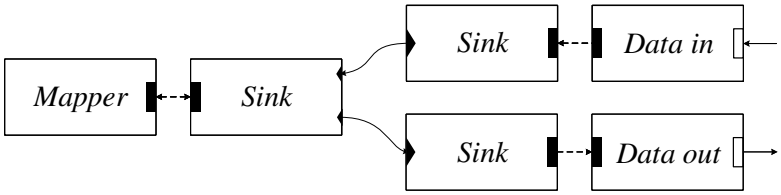


Figure 6.3: The different types of adapters in a sample configuration. The central adapter is the Mapper. This is a processing adapter that forwards data from input adapters to processing adapters and to output adapters.

used to feed data from a network source to the platform. An output agent port, on the other hand, can be used to forward data to another platform.

Figure 6.2 shows the layout of an adapter.

6.4.1 Adapter Types

FLINT supports three types of adapters—*I/O* adapters, *processing* adapters and *direct* adapters. I/O adapters use their agent port to fetch data from a data source or push data to a destination. *Processing* adapters, on the contrary, do not use their agent ports and return the data to the sink after processing. Figure 6.3 shows how the different types of adapters work in a simple configuration. Data coming from a data source is captured using the input port of the *Data in* agent. The corresponding element forwards the data to the *Mapper* adapter. This is a mandatory processing adapter responsible for data delivery to the next adapter in the chain. Incoming data are passed to the agent and distributed over one or more outputs. The *Data out* I/O adapter can perform extra processing to match the format of the data source it is connected to or immediately forward the data to the destination.

Some configurations require bidirectional communication. I/O adapters therefore support data flowing in both directions. Their corresponding sink element is subscribed to a unique topic for communication happening between adapters. Communication coming from an external data source is published to the central *Mapper* adapter acting as a router. This allows for fine-grained configuration in both the upward and downward direction.

Finally, FLINT also supports the use of *Direct* adapters. These are a special type of adapter that omit the *Mapper* adapter. This can be useful to connect a fleet of devices directly to an adapter chain.

6.4.2 Device Based Context

FLINT uses the *Mapper* adapter as a hub for data distribution. The Mapper is a simple processing adapter with multiple sink inputs and outputs. Incoming data are processed on a per-device basis. The agent matches a device's Medium Access Control (MAC) address with an adapter based on the input topic. A routing scheme is constructed from the configured information for a particular device. The sink then forwards the original payload and the routing scheme to the next adapter in the chain. From there, based on the routing scheme, the processed packet is distributed to the next hop.

Adapters have a unique identifier. These identifiers are used to construct chains of adapters. Every adapter, except for the Mapper adapter, subscribes to their input topic, i.e., *urn:uuid:adapter-uuid/in*. The Mapper, on the contrary, is subscribed to the output topic of every adapter, i.e., *+/out*.

Figure 6.4 shows a directed graph using the different types of adapters. The *io-1* and *io-2* adapters are I/O adapters. Adapter *p-1* is a processing adapter and *d-1* is a direct adapter. In this example, adapter *io-1* publishes its input data to the output topic *urn:uuid:io-1/out* to request a routing scheme from the mapper. An exact match for a device's MAC address and the requesting adapter will return the configured routing scheme. The packet from adapter *io-1* is sent to the next hop in the chain, i.e., *p-1*. From there, *p-1* forwards the modified data to the next adapter in the chain—*io-2*. At this point, data are output using the adapter's agent. Similarly, communication flowing in the other direction is published to the adapter's output topic to request routing information from the mapper. The scheme in the other direction excludes *p-1* from the chain and is directly forwarded to *io-1*. Direct adapters, such as *d-1*, forward data directly to their destination. This can be useful to serve a fleet of devices.

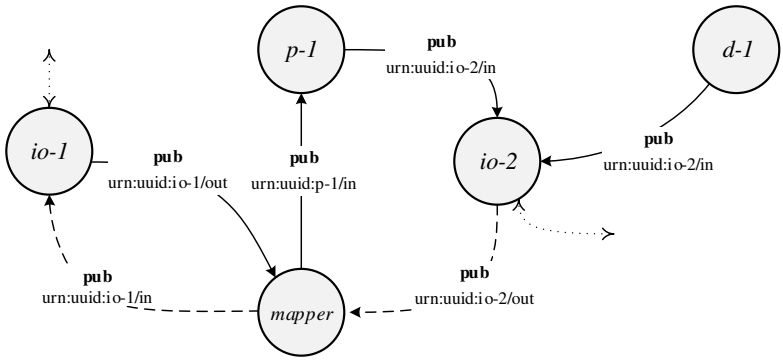


Figure 6.4: A directed graph constructed using the different types of adapters. Adapter io-1 is an I/O adapter. Adapter p-1 is a processing adapter and adapter io-2 is another I/O adapter. The mapper adapter passes the messages along the vertices. Adapter d-1 communicates directly with io-2.

6.4.3 Packet Storage

FLINT is built from the observation that IoT devices can have multiple network interfaces that must be routed to multiple outputs. These devices can also have restricted communication opportunities due to their energy savings mechanisms. Therefore, the *Mapper* adapter keeps track of the last active network for every device and their corresponding downlink scheme. Traffic flowing in the downward direction will be routed towards the active network interface or will be queued when no interface is active. Consequently, other FLINT adapters do not require the implementation of queues between adapters or any other routing logic. The Mapper adapter forwards a routing scheme upon a request on its input, which can be interpreted by the *sink* from any other adapter. Hence, the *Mapper* adapter is a mandatory component in more advanced configurations.

6.4.4 Device Configuration

FLINT device configurations are written in JavaScript Object Notation (JSON), based on the W3C Thing Description (TD) ontology. A standard TD is extended using three keys: `IPv6`, `ioAdapterDefinitions` and `adapterScheme`. `ioAdapterDefinitions` contain values required for bidirectional communication. The `adapterScheme` indicates how the different adapters are connected. An example is given in Listing 6.1.

```
{
  "id": "urn:5f3cc650-c91a-477f-9726-f5f27097b6c7",
```

```
"@context": "https://www.w3.org/2019/wot/td/v1",
"title": "SampleDevice",
"ipv6": ["2001:0db8:85a3:0000:0000:8a2e
   :0370:7334"],
"ioAdapterDefinitions":
[{
  "uuid": "urn:uuid:cc0c1e3b-ce09-4c44-b9da-57
     ee9871497a",
  "deviceDefinitions": {
    "interfaceType": "continuous"
    "mac": "0004a30b0024e96c"
  }
},
{
  "uuid": "urn:uuid:489edd56-4b1c-4332-a5ba-
     cad14994668d",
    "deviceDefinitions": {
      "interfaceType": "uplink_triggered",
      "mac": "92BC10"
    }
},
{
  "uuid": "urn:uuid:1323b31a-1af7-4548-aad7-89
     f45f7b5713",
    "deviceDefinitions": {
      "id": "2"
    }
}],
"adapterScheme": [
[
  ["urn:cc0c1e3b-ce09-4c44-b9da-57ee9871497a", "
     urn:uuid:489edd56-4b1c-4332-a5ba-
     cad14994668d"]
  ["urn:uuid:8b6092c2-09a4-4aca-aca2-2948195c10de
     "],
  ["urn:uuid:1323b31a-1af7-4548-aad7-89f45f7b5713
     "]
],
[
  ["urn:uuid:1323b31a-1af7-4548-aad7-89f45f7b5713
     "],
  ["urn:cc0c1e3b-ce09-4c44-b9da-57ee9871497a", "
     urn:uuid:489edd56-4b1c-4332-a5ba-
     cad14994668d"]
]
```

```
  ],
  ...
}
```

Listing 6.1: A Thing Description contains a list of adapter definitions. These adapters can be used in a scheme to construct a directed graph.

An `ioAdapterDefinition` contains routing information. Every definition can be linked to an adapter using its uuid. The *Mapper* adapter also requires the MAC address of every I/O interface to uniquely target a device and manage the queue based on the `interfaceType`. These types can be divided in three groups. A `continuous` interface can be reached at any given moment. A `beacon` interface can be interfaced with during predefined intervals and an `uplink_triggered` interface can be reached only after an uplink transmission.

The `adapterScheme` is a JSON array constructed of *chains* that contain *shackles*, which on their turn contain a list of adapters. A chain starts with an array of I/O adapters (a shackle). Every adapter in this array can be a possible source of data. The *Mapper* adapter will therefore keep track of the last active adapter in this shackle. The chain ends with a shackle of I/O adapters. Processing adapters can be added at any level in between them. Every shackle in the chain indicates a hop. An adapter that receives a packet on its input will forward the processed packet to *every* adapter in the next hop.

The example `adapterScheme` in Listing 6.1 contains two chains. The first chain is used for traffic flowing in the upward direction and contains three shackles. The first shackle contains two adapters that are used to interface with the device directly. Data from any of these adapters will flow to the adapter in the next shackle that contains a processing adapter (urn:uuid:8b6092c2-09a4-4aca-aca2-2948195c10de). The last shackle in the chain will output the processed data to a given destination. The second chain contains the inverted configuration, used for communication in the **downward** direction. In this example, the processing adapter is not required for communication in this direction and is removed from the chain. The last shackle again contains the two adapters that can interface directly with the device. The *Mapper* adapter keeps track of the last active adapter to forward data to.

Communication between the adapters also happens in JSON. Every adapter must therefore adhere to a JSON scheme to provide syntactical interoperability.

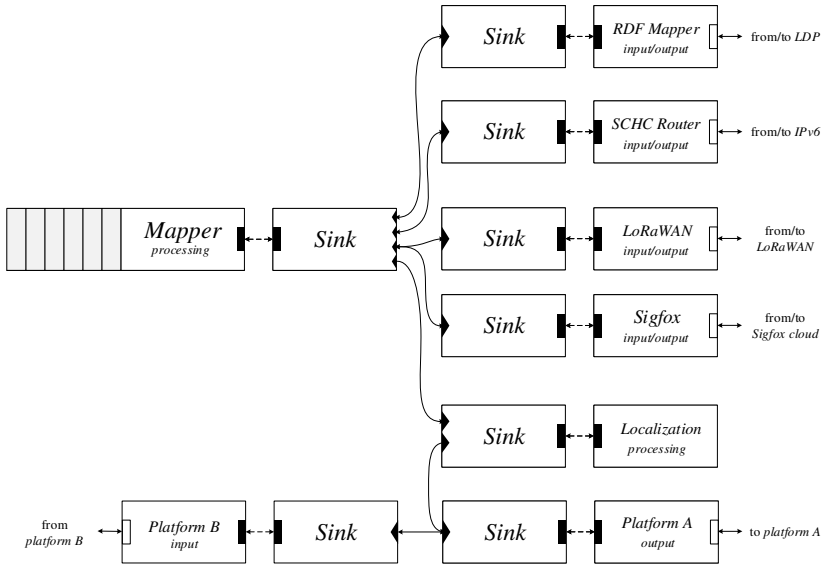


Figure 6.5: More complex system configuration. Every sink is connected to others via the message bus. The Mapper adapter implements a queue.

6.5 Evaluation

This section evaluates a real FLINT configuration: two Low Power Wide Area Networks (LoRaWAN and Sigfox) serve as a communication technology for several low power devices. Some devices use the novel SCHC standard to compress IPv6 packets. Others require enriching their data with LPWAN localization before forwarding it to an IoT platform (e.g., Linked Data Platform (LDP), Obelisk). An overview of the complete system configuration is given in Figure 6.5.

The system is evaluated in terms of interoperability, scalability and performance.

6.5.1 Levels of Interoperability

Interoperability in the IoT can be seen from different perspectives such as syntactic interoperability, networking interoperability, device interoperability, semantic interoperability, and platform interoperability [4]. These subdivisions are further explained in the following sections while being applied to FLINT.

6.5.1.1 Syntactic Interoperability

Processing tasks that involve local information do not require any adaptation to the network. The SCHC router, for example, must store information about the fragmentation state of a device. If the device does not acknowledge a fragment in time, the SCHC router will re-transmit the fragment or discard the packet. These actions depend on the packet's content and do not involve any other adapters. Other processing tasks, however, do require information from another adapter in the chain.

Passing information between adapters requires a common messaging pattern, often referred to as syntactic interoperability. FLINT uses a message scheme that adapters must adhere to. The message scheme contains fields to carry information along and may include:

- *Device information.* Information such as the device MAC address and the original data packet are stored in this field. Any additional information can be added to the `device-custom-ctrl` field.
- *Adapter information.* The `adapter-ctrl` field contains information about the previous adapter in the chain. Information about the chain is also stored in this field. Every I/O adapter first consults the *Mapper* adapter to generate the routing scheme that is added to this field.
- *Input information.* The `input-ctrl` field contains information about the adapter that first received the packet. Custom information about the network the adapter is connected to, can be added to the field `input-custom-ctrl`.
- *Output information.* Additional information about the device is stored in the `output-ctrl` field. The TD of every device contains the uuid of each device and is added to the message by the *Mapper* adapter. Other adapters in the chain must be able to uniquely identify devices, regardless of their input adapter. The uuid provides an abstraction for devices that have multiple network interfaces.

Every *sink* serializes these messages using the above grammar in order to provide *syntactical interoperability.*

6.5.1.2 Device and Network Interoperability

The lowest level of interoperability that FLINT can guarantee, is device and network interoperability. Device interoperability ensures that *high-end IoT devices*, such as smartphones, can communicate with resource constrained, *low-end IoT devices* [4]. The *Mapper* interface provides a queue and supports

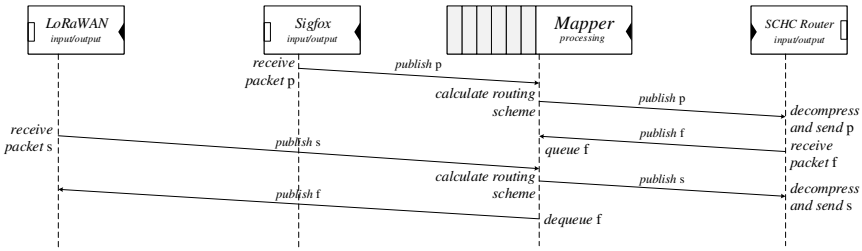


Figure 6.6: Mobility and queue management in a FLINT system. This diagram shows how packets travel through a FLINT system configuration. Time moves downwards. The central element is a Mapper adapter. Packets are distributed from transmitting devices to the destination. Packets in the downward direction are queued and dequeued according to the device's `interfaceType`.

bidirectional communication over multiple network interfaces. Both *low-end devices* and *high-end devices* can exchange information using different communication technologies with diverse downlink patterns. Figure 6.6 shows an excerpt of the complete FLINT configuration that provides device and network interoperability for a *low-end device* with a Sigfox and LoRaWAN interface. The Sigfox adapter forwards packet p together with the device's MAC address to the *Mapper* adapter. The routing scheme is calculated based on the information in the TD. Packets coming from these networks are SCHC compressed and forwarded to the SCHC router. This adapter handles requests to, and responses from, the IPv6 network. The response from the IPv6 network (packet f) is stored in the queue of the *Mapper* adapter, since downward traffic for these devices can not flow continuously. Finally, due to a network change, the device transmits packet s over its LoRaWAN interface. The *Mapper* adapter will immediately dequeue packet f and forward it to the corresponding LoRaWAN adapter. Packet s is also delivered to the SCHC router.

As illustrated, the *Mapper* adapter is required to provide *device and network interoperability*.

6.5.1.3 Semantic and Platform Interoperability

FLINT is well suited to provide both semantic and platform interoperability. For example, imagine a LPWAN sensor network with the following requirements:

- *Low-end* sensors must be manageable through the LwM2M protocol. Preferably, their packets should be compressed using the SCHC compression and fragmentation standard.

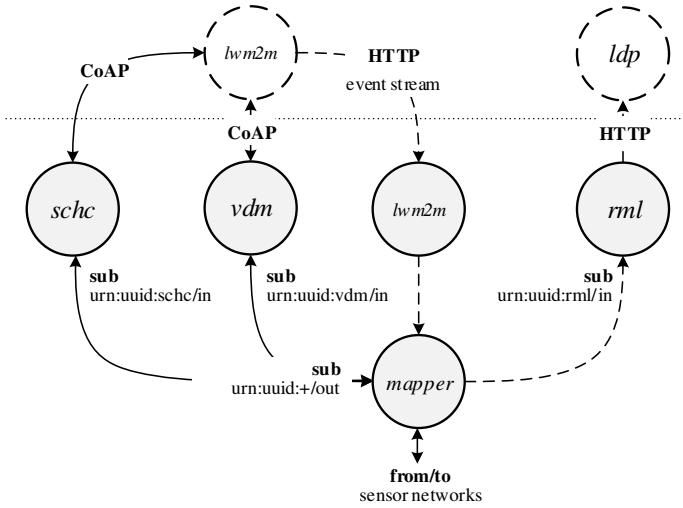


Figure 6.7: FLINT configuration that provides semantic and platform interoperability. Packets from the LPWANs are delivered to the LwM2M server. Data from the LwM2M server are mapped to an ontology that matches the semantics of the LDP.

- For proprietary sensors, an adapter must run as a digital twin in the FLINT platform. This can be done by converting their proprietary format using the Virtual Device Manager (VDM) [29].

- Data from the LwM2M server must be delivered to a LDP. Hence, the LwM2M ontology must be mapped to RDF. This can be done using the RML.

FLINT's modular and extensible architecture makes this easy; Figure 6.7 shows the configuration. Data coming from the LPWANs are delivered either to the SCHC router or the VDM. Both adapters send requests to the LwM2M server and deliver responses to the *Mapper* adapter. The LwM2M adapter subscribes to the event stream of the LwM2M server. The RML adapter on its turn maps the LwM2M ontology to the Semantic Sensor Networks (SSN) ontology. The result is published to a LDP.

The above configuration shows how two incompatible platforms can be interconnected through FLINT. An adapter is dedicated to match their semantics and thereby provides *semantic and cross-platform interoperability*. Once cross-platform interoperability is achieved, *cross-domain interoperability* can be enabled by adding adapters that integrate platforms from heterogeneous domains.

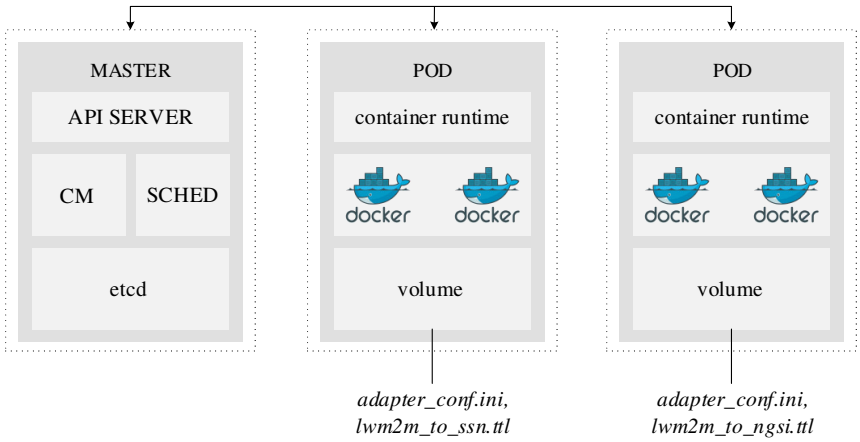


Figure 6.8: An excerpt of a FLINT system in a Kubernetes environment. Every Kubernetes deployment consists of (at least) two Docker containers—the sink and the agent. Both containers can be configured by mounting configuration files. Every deployment is managed by the Kubernetes master.

6.5.2 Scalability

This section evaluates FLINT in terms of scalability. Every FLINT adapter consists of at least two components—the agent and the sink. Consequently, every adapter consists of (at least) two containers. Components can be updated easily when pulling an image in a Kubernetes deployment. Kubernetes thereby provides a fast iteration cycle and scalability. Consider as an example the data flow from Section 6.5.1.3.

There, a FLINT configuration uses RML to match different ontologies. The RML adapter uses a Turtle mapfile that describes the mapping. The Kubernetes configuration file is used to input the mapfile to the RML adapter that runs in a Node.js server instance. In a running system, this adapter can be replicated so multiple adapters can match the LwM2M vocabulary to various ontologies. This can be done by simply replacing the mapfile. Figure 6.8 shows two Kubernetes deployments that are managed by the Kubernetes master. A Kubernetes deployment represents an adapter and can be configured by mounting configuration files. In this example, the LwM2M data model can be translated to SSN by one adapter and to Next Generation Service Interfaces-Linked Data (NGSI-LD) models, such as the Port model, by another one. By simply replicating the adapter and mounting a different configuration file, the FLINT configuration can be expanded easily.

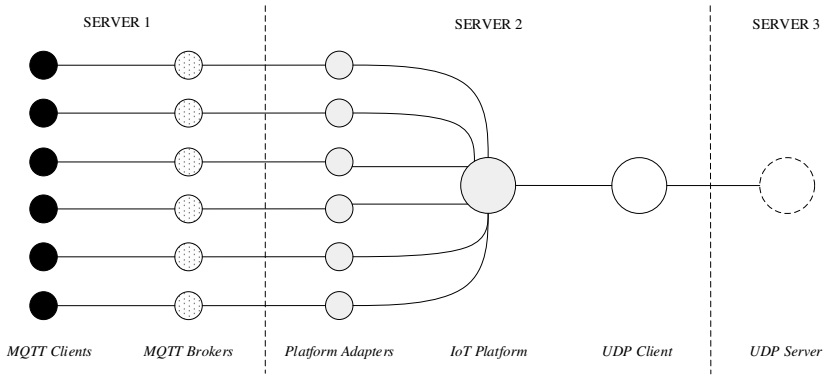


Figure 6.9: The test setup. Every test consists of six MQTT clients, subscribed to six MQTT brokers. The clients forward their data to the IoT platform that is being tested. A UDP client fetches data from the platform and forwards the packet to a UDP server.

6.5.3 Performance Evaluation

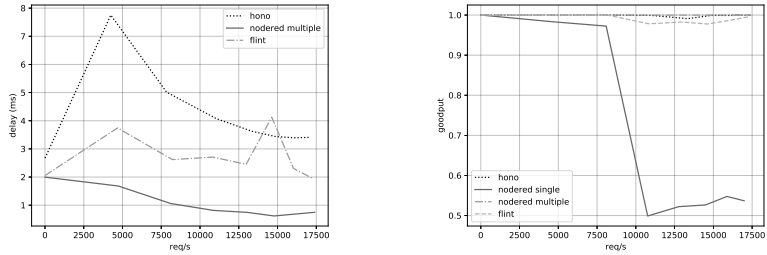
This section first compares FLINT with other State of the Art (SOTA) platforms that provide similar functionality (i.e., open source N*-N*-N* platforms). The evaluation uses six MQTT clients that publish data to a broker. For every platform, a platform adapter processes the incoming data in order to feed it to the IoT platform. The platform delivers the data to a UDP client that forwards the data to a UDP server. This is illustrated in Figure 6.9.

6.5.3.1 Experimental Setup

The experimental setup consists of three servers running Ubuntu 18.04 and are synced using the Network Time Protocol daemon (ntpd). All three servers have a gigabit Network Interface Card (NIC), 2 Hexacore Intel E5645 (2.4GHz) Central Processing Unit (CPU) and 24 Gigabyte (GB) Random Access Memory (RAM). One server is used to host the MQTT clients and brokers. Another to host the platforms and one to host the UDP server. Each IoT platform is configured to run in a Kubernetes cluster.

6.5.3.2 Analysis of Platform Performance

This section analyzes the different selected platforms under different loads. The responsiveness of every platform is measured using the goodput. This has been defined as the total number of successfully received packets divided by the total number of sent packets. Figure 6.10 shows the average time and



(a) Average delay in seconds (b) Goodput

Figure 6.10: The latency (in seconds) and goodput measured for six MQTT adapters in different platforms. Multiple instances of Node-RED perform best compared to Hono and FLINT. Node-RED single had too large delays for the Figure, but can be retrieved from Table 6.2.

goodput for Node-RED (v.1.3.5), Eclipse Hono (v 1.9) and FLINT.

Excluded from the Figure are the very large delays for a single Node-RED instance. For completeness, these values are provided in Table 6.2. These high delays are due to the single-threaded design of the underlying Node.js instance [30]. Furthermore, the goodput for a Node-RED (single instance) application dropped significantly for data rates above 7500 messages per second, which can be seen in Figure 10b. To take advantage of multi-core systems, a cluster of Node.js processes can be deployed to handle the load. Therefore, six Node-RED instances were deployed and is referenced as *Node-RED multiple*. Every instance subscribes to a MQTT broker, processes the data and forwards it to the UDP server. This resulted in near-zero latency and a 100% goodput.

For this experiment, Hono used the default configuration. However, as Hono seemed to struggle with data rates above 5000 messages/second (which was also reported in [31]), we deployed a Vert.x MQTT adapter for every MQTT client. This resulted in slightly worse results in terms of latency and goodput, compared to Node-RED.

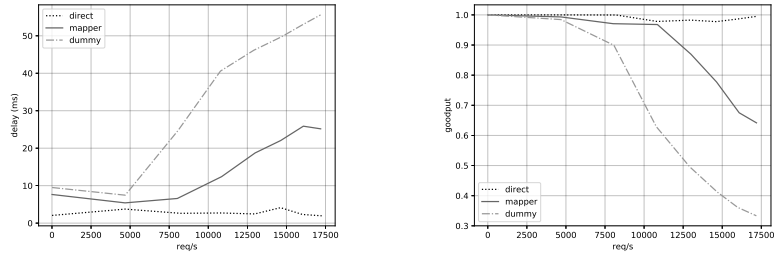
Finally, FLINT consisted of six *direct* MQTT adapters that published their data directly over the message bus to the UDP I/O adapter. Even though FLINT requires at least twice the payload size of the other platforms, the experiment showed slightly better results compared to Hono and slightly worse results than Node-RED running multiple instances. We noticed that the MQTT broker was dropping packets, which could have caused the 1% packet loss. Increasing the queue size of the broker resulted in 0% packet loss, however did increase the latency tremendously. The broker could in

this case act as some sort of shock buffer and does not necessarily explain the real cause of packet loss.

6.5.4 Mapper Forwarding Rate

FLINT provides flexibility in connecting non-interoperable IoT networks. Due to its modularity, it is possible to interconnect various IoT components on top of which the *Mapper* adapter can queue packets based on the interface type of the device. These advantages, however, come at a cost. This section analyzes the drawbacks introduced by the platform.

The above section used six *direct* adapters. This type of adapter forwards the data directly to another endpoint in the platform. However, the *Mapper* adapter can be used to forward data between several components. As shown in Figure 6.11, the average latency increases and the goodput decreases, since the message bus must process twice as many messages.

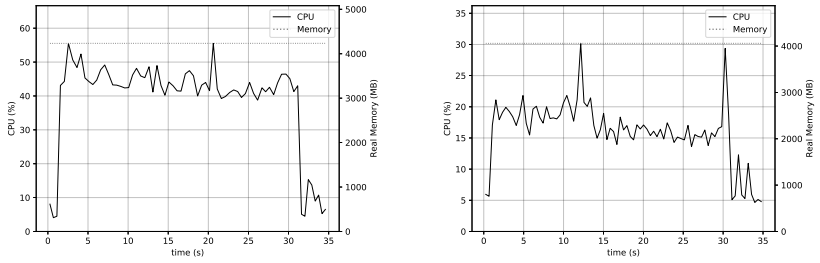


(a) Average delay in seconds (b) Goodput

Figure 6.11: The latency (in seconds) and goodput for six MQTT adapters measured for different configurations in FLINT.

In order to measure the impact of another hop, a *dummy* adapter was added to the system. This is a processing adapter that forwards the packet after a *No Operation* (NOP). Messages will now flow from the MQTT adapter to the *Mapper* adapter, to the *dummy* adapter and finally to the UDP adapter. Using this configuration, the message bus must process three times the amount of messages compared to the second configuration.

An overview of the measurements is given in Table 6.2. It can be seen that the message bus and the *Mapper* adapter form a bottleneck for the system. In the future, this can be solved in several ways. A first solution could use MQTT v5 Shared Subscriptions. This feature distributes the load across all subscribers on the same topic and is sometimes referred to as client load balancing. This allows setting up multiple Mapper adapter instances without having duplicates. To avoid saturation of the MQTT broker, a cluster of MQTT brokers can be deployed. Managing a cluster of MQTT



(a) FLINT Direct adapter configuration without load balancing

(b) FLINT Direct adapter configuration with external broker

Figure 6.12: Comparison of a very simple load balancing method. The first configuration runs every component on the same machine, while in the second configuration the broker is moved to a different machine.

brokers, however, requires non-standard broker discovery, subscriber takeover and routing management. Therefore, other messages busses, such as Apache Kafka or Pulsar [32] can be used. This would require a few changes to the system. First, the sink must be replaced in order to communicate with the improved message bus. In Apache Kafka, multiple partitions can then be used to form consumer groups. These groups avoid duplication of data over multiple members subscribed to the same topic. As such, multiple mapper instances can be deployed, without the need to redesign the Mapper adapter completely. However, all Mapper adapters should access the same Thing Descriptions, and therefore require access to a shared Thing Directory.

6.5.5 Resource Consumption

Finally, in Figure 6.12, two system configurations are presented in order to explore the impact of a very basic workload offloading approach. In Figure 6.12, a, every component runs on the same machine. This results in an average of around 45% CPU load and 4.5 GB memory consumption. In the second configuration, the MQTT broker was moved to a different machine. This resulted in an average of only 17% CPU usage and 4 GB memory consumption.

This confirms the previous statement that the message bus forms a bottleneck for the system, since every message between components passes through the broker. However, due to FLINT's modularity, the message bus can be transferred easily to a different system in order to divide the load, at the cost of a higher network delay.

Table 6.2: The Latency (L) in seconds, the Goodput (G) and the Payload size in bytes for every platform using different configurations.

req/s	**4500**		**10500**		**14500**		**17500**		**All**
	L (s)	*G*	*L (s)*	*G*	*L (s)*	*G*	*L (s)*	*G*	*Payload size (B)*
Node-RED (1)	0.007	0.98	11.633	0.5	20.843	0.53	26.278	0.54	259
Node-RED (6)	0.0016	1	0.0008	0.99	0.0006	1	0.0007	0.99	259
Eclipse Hono	0.007	1	0.004	0.99	0.003	0.99	0.003	0.99	259
FLINT (direct)	0.003	1	0.002	0.98	0.004	0.98	0.002	0.99	513
FLINT (mapper)	0.005	0.99	0.012	0.97	0.022	0.78	0.025	0.64	1102
FLINT (dummy)	0.007	0.98	0.04	0.63	0.05	0.41	0.054	0.33	1202

6.6 Discussion

Other platforms, such as INTER-IoT and Eclipse node-wot have been around to provide interoperability in the IoT. However, INTER-IoT has become a rather complex platform that provides multi-layer interoperability for large scale projects. Eclipse node-wot, implemented in Node.js, merely focuses on interoperability, rather than scalability.

These observations led to the initial design of FLINT with the main objective to interconnect non-interoperable IoT networks that are characterised by intermittent connectivity. FLINT was designed to support five levels of interoperability—syntactic interoperability, semantic interoperability, device interoperability, network interoperability and platform interoperability. The modularity of the design, consisting of a *sink* and *agent*, allows the recycling of existing programs through the use of a socket connection, while still being able to scale.

We implemented FLINT to support our design choices and to show how the implementation behaves against different loads on comparable platforms. Our evaluation showed that Node-RED performs best in terms of latency and goodput. However, this requires to break apart the application in order to take advantage of the available CPU cores of the system. Eclipse Hono and FLINT show similar results. However, due to its flexible design using the *sink*, developers require only little understanding of the underlying protocols and can use any language that supports UDP sockets. Integrating business logic in Eclipse Hono, on the contrary, requires both an AMQP library and understanding of the protocol.

The evaluation showed that FLINT is able to deliver good performance in a scalable, container based system. However, both the performance evaluation of the *Mapper* adapter and the resource consumption of the basic load balancing approach show the impact of a single, central message bus. Since every message has to pass through the broker and, if configured, the *Mapper* adapter, this forms a bottleneck for the system. Therefore, other distributed streaming platforms, such as Apache Kafka or Pulsar can be used in the future to provide support for use cases that require higher data rates.

6.7 Conclusions

FLINT is an open and extensible distributed platform architecture. Chains of adapters can be built in order to serve non-interoperable IoT devices. FLINT was developed and tested in the scope of the Port of the Future where several adapters interconnect networks and platforms. Due to the modularity of an

adapter, developers can easily integrate existing software while maintaining a flexible design. This will allow further evolution of separate networks and platforms and may contribute to inter-port interaction, port-city interaction and trans-sector innovation in general. Our performance analysis shows that the modularity is comparable to other existing platforms, while offering the ability to develop software using rapid prototyping approaches. In order to maintain a scalable platform, components can be replicated in a distributed network. However, scalability should still be looked at in more detail. Duplicates must be avoided when deploying multiple adapter instances and the abilities of different messages busses should be analyzed. FLINT is free software; it is available for download at https://github.com/imec-idlab/flint (accessed on 06/10/2021).

References

[1] Amazon. *AWS IoT Core for LoRaWAN*. Available from: https://docs.aws.amazon.com/iot/latest/developerguide/connect-iot-lorawan.html.

[2] K. Technologies. *Kaa Enterprise IoT Platform*. Available from: https://www.kaaiot.com.

[3] M. Inc. *ThingSpeak Internet of Things*. Available from: https://thingspeak.com/.

[4] M. Noura, M. Atiquzzaman, and M. Gaedke. *Interoperability in Internet of Things: Taxonomies and Open Challenges*. Mobile Networks and Applications, 24(3):796–809, June 2019. Available from: http://link.springer.com/10.1007/s11036-018-1089-9, doi:10.1007/s11036-018-1089-9.

[5] Y. Yang, M. Zhong, H. Yao, F. Yu, X. Fu, and O. Postolache. *Internet of things for smart ports: Technologies and challenges*. IEEE Instrumentation & Measurement Magazine, 21(1):34–43, February 2018. Available from: http://ieeexplore.ieee.org/document/8278808/, doi:10.1109/MIM.2018.8278808.

[6] T. Inkinen, R. Helminen, and J. Saarikoski. *Port Digitalization with Open Data: Challenges, Opportunities, and Integrations*. Journal of Open Innovation: Technology, Market, and Complexity, 5(2):30, May 2019. Available from: https://www.mdpi.com/2199-8531/5/2/30, doi:10.3390/joitmc5020030.

[7] J. Famaey, R. Berkvens, G. Ergeerts, E. D. Poorter, F. V. D. Abeele, T. Bolckmans, J. Hoebeke, and M. Weyn. *Flexible Multimodal Sub-Gigahertz Communication for Heterogeneous Internet of Things Applications*. IEEE Communications Magazine, 56(7):146–153, jul 2018. doi:10.1109/MCOM.2018.1700655.

[8] V. Bracke, M. Sebrechts, B. Moons, J. Hoebeke, F. De Turck, and B. Volckaert. *Design and evaluation of a scalable Internet of Things backend for smart ports*. Software: Practice and Experience, 51(7):1557–1579, July 2021. Available from: https://onlinelibrary.wiley.com/doi/10.1002/spe.2973, doi:10.1002/spe.2973.

[9] B. Buurman, J. Kamruzzaman, G. Karmakar, and S. Islam. *Low-Power Wide-Area Networks: Design Goals, Architecture, Suitability to Use Cases and Research Challenges*. IEEE Access, 8:17179–17220, 2020. doi:10.1109/ACCESS.2020.2968057.

[10] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and JC. Zúñiga. *SCHC: Generic Framework for Static Context Header Compression and Fragmentation.* Technical Report RFC8724, RFC Editor, April 2020. Available from: https://www.rfc-editor.org/info/rfc8724, doi:10.17487/RFC8724.

[11] B. Moons, A. Karaagac, J. Haxhibeqiri, E. D. Poorter, and J. Hoebeke. *Using SCHC for an optimized protocol stack in multimodal LPWAN solutions.* In 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), pages 430–435, Limerick, Ireland, April 2019. IEEE. Available from: https://ieeexplore.ieee.org/document/8767210/, doi:10.1109/WF-IoT.2019.8767210.

[12] M. Aernouts, N. BniLam, N. Podevijn, D. Plets, W. Joseph, R. Berkvens, and M. Weyn. *Combining TDoA and AoA with a particle filter in an outdoor LoRaWAN network.* In 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), pages 1060–1069. IEEE, apr 2020. doi:10.1109/PLANS46316.2020.9110172.

[13] T. Janssen, M. Weyn, and R. Berkvens. *A Primer on Real-world RSS-based Outdoor NB-IoT Localization.* In 2020 International Conference on Localization and GNSS (ICL-GNSS), pages 1–6. IEEE, jun 2020. doi:10.1109/ICL-GNSS49876.2020.9115578.

[14] G. G. Anagnostopoulos and A. Kalousis. *A Reproducible Analysis of RSSI Fingerprinting for Outdoor Localization Using Sigfox: Preprocessing and Hyperparameter Tuning.* In 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pages 1–8. IEEE, sep 2019. arXiv:1908.06851, doi:10.1109/IPIN.2019.8911792.

[15] F. Zafari, A. Gkelias, and K. K. Leung. *A Survey of Indoor Localization Systems and Technologies.* IEEE Communications Surveys & Tutorials, 21(3):2568–2599, 2019. doi:10.1109/COMST.2019.2911558.

[16] Y. Li, Y. Zhuang, X. Hu, Z. Gao, J. Hu, L. Chen, Z. He, L. Pei, K. Chen, M. Wang, X. Niu, R. Chen, J. Thompson, F. Ghannouchi, and N. El-Sheimy. *Toward Location-Enabled IoT (LE-IoT): IoT Positioning Techniques, Error Sources, and Error Mitigation.* IEEE Internet of Things Journal, 4662(c):1–1, 2020. arXiv:2004.03738, doi:10.1109/JIOT.2020.3019199.

[17] M. Aernouts, F. Lemic, B. Moons, J. Famaey, J. Hoebeke, M. Weyn, and R. Berkvens. *A Multimodal Localization Framework Design for IoT Applications.* Sensors, 20(16):4622, aug 2020. doi:10.3390/s20164622.

[18] *Data - W3C*. Available from: https://www.w3.org/standards/semanticweb/data.

[19] A. Dimou, M. V. Sande, and P. Colpaert. *RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data*. page 5, 2014.

[20] S. Sinche, D. Raposo, N. Armando, A. Rodrigues, F. Boavida, V. Pereira, and J. S. Silva. *A Survey of IoT Management Protocols and Frameworks*. IEEE Communications Surveys Tutorials, 22(2):1168–1190, 2020. Conference Name: IEEE Communications Surveys Tutorials. doi:10.1109/COMST.2019.2943087.

[21] O. LwM2M. *Lightweight Machine to Machine Technical Specification: Core*, November 2020. Available from: http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf.

[22] O. Foundation. *Node-RED: Low-code programming for event-driven applications*. Available from: https://nodered.org/.

[23] Thingweb. *ThingWeb: A Web of Things Implementation*. Available from: https://www.thingweb.io/.

[24] P. Gimenez and M. Llop. *Interoperability of IoT platforms in the port sector*. other, March 2020. Available from: https://www.morressier.com/article/5e4fe9bf6bc493207536f6d6, doi:10.26226/morressier.5e4fe9bf6bc493207536f6d6.

[25] O. Inc. *OpenRemote: The 100% Open Source IoT Platform*. Available from: https://openremote.io/.

[26] G. Fortino, C. Savaglio, C. E. Palau, J. S. de Puga, M. Ganzha, M. Paprzycki, M. Montesinos, A. Liotta, and M. Llop. *Towards Multilayer Interoperability of Heterogeneous IoT Platforms: The INTER-IoT Approach*. In R. Gravina, C. E. Palau, M. Manso, A. Liotta, and G. Fortino, editors, Integration, Interconnection, and Interoperability of IoT Systems, pages 199–232. Springer International Publishing, Cham, 2018. Series Title: Internet of Things. Available from: http://link.springer.com/10.1007/978-3-319-61300-0_10, doi:10.1007/978-3-319-61300-0_10.

[27] T. T. Authors. *ThingsBoard: Open-source IoT Platform*. Available from: https://thingsboard.io/.

[28] T. E. H. Project. *Eclipse Hono: Connect, Command & Control IoT devices*. Available from: https://www.eclipse.org/hono/.

[29] A. Karaagac, M. VanEeghem, J. Rossev, B. Moons, E. DePoorter, and J. Hoebeke. *Extensions to LwM2M for Intermittent Connectivity and Improved Efficiency.* In 2018 IEEE Conference on Standards for Communications and Networking (CSCN), pages 1–6, Paris, France, October 2018. IEEE. Available from: https://ieeexplore.ieee.org/document/8581821/, doi:10.1109/CSCN.2018.8581821.

[30] *Cluster | Node.js v16.5.0 Documentation.* Available from: https://nodejs.org/api/cluster.html.

[31] J. Reimann. *We scaled IoT – Eclipse Hono in the lab*, July 2018. Available from: https://dentrassi.de/2018/07/25/scaling-iot-eclipse-hono/.

[32] V. John and X. Liu. *A Survey of Distributed Message Broker Queues.* page 9.

7
Conclusion

"A dream you dream alone is only a dream. A dream you dream together is reality."

John Lennon

Interoperability is something that has been part of the Internet since the American Ministry of Defence introduced the Advanced Research Projects Agency Network (ARPANET) in 1969. Researchers proposed packet switching methods in Request For Comments (RFC) to provide a common way of communicating. The foundations of the Web were also laid by Tim Berners-Lee's proposal about standardized information sharing. Without these standardization efforts both the World Wide Web (WWW) and the Internet wouldn't have contributed to the largest communication platform in human history. Private companies are nevertheless often looking to provide the highest efficiency possible and consequently proprietary solutions have been around since the development of the first terrestrial communication systems. However, standardization is much needed to provide trade-offs between efficiency and interoperability so that global communication systems can sprout.

Just like the Internet and the WWW grew out of a connection of machines, so will the Internet of Things (IoT) emerge from a connection of smart devices. In order to connect these IoT devices, many wireless technologies

are available. Communication protocols such as IEEE 802.15.4, ZigBee and Bluetooth Low Energy (BLE) provide wireless links for short range IoT devices, while technologies such as Long Range (LoRa) and Sigfox, provide very long range wireless communication links. These Low Power Wide Area Network (LPWAN) technologies provide a promising solution for many use cases ranging from Smart Logistics to Smart Ports. Energy savings, structure monitoring and asset tracking are just a few of the many cases that can benefit from the added efficiency that the IoT is expected to bring. Their long range and low power consumption provide a communication link for devices that should operate on a single battery charge for multiple years. However, these benefits come at the cost of very low bandwidth. These constraints complicate the issue of interoperability in this domain. Not only will interoperable systems limit the integration effort and speed up the adoption of these technologies, they can also integrate them into the Web of the future. As has been proven in the past a single smart device cannot add much value, it is the combination of things (i.e. an ecosystem) that will disrupt the future.

Therefore, this PhD examined interoperability issues from sensor to storage in LPWANs. Not much research has been conducted to provide a complete interoperable LPWANs system that can be plugged into future versions of the Internet and the WWW. In order to connect these devices to the Internet following its well-proven end-to-end principle, this dissertation investigated the novel Static Context Header Compression (SCHC) protocol. This adaptation layer sits between the Medium Access Control (MAC) layer and the network layer and compresses the upper layers of the Internet protocol stack in order to limit the overhead over these low bandwidth technologies. Current solutions lack the ability to provide an interoperable solution which, as shown in Chapter 2, can be solved by SCHC.

However, given the static nature of this protocol, the current solution cannot solve the end-to-end issue entirely. LPWANs typically connect via an Application Programming Interface (API) to a back-end application, resulting in an engineered link for every technology as we showed in Chapter 4. Therefore, Chapter 3 provides a way to move the complexity in such networks to the gateway. Both solutions result in a higher energy consumption due to the extra traffic to complete the registration flow. Even though the specification has been designed following the fact that the traffic flows to be compressed are known in advance, the slightest change in the application will result in a significant higher energy consumption. This was shown on the basis of an example taken from the Light Weight Machine to Machine (LwM2M) specification. LwM2M has a way to provision devices with the credentials of the network and thus requires altering the context during operation. We

showed that in such a situation a constrained device can benefit from both solutions.

The work done in Chapter 3 provided a solution to move away from systems that can be integrated only by using their APIs. However, in order to incorporate multiple technologies on a single device, a central orchestration entity will always be required to keep track of the last active gateway. Therefore, a modular architecture was proposed in Chapter 4 to handle transmissions over multiple technologies. In such architecture, LPWAN technologies can complement each other. Medium range technologies can be used to distribute Over The Air (OTA) Updates (OTAUs), offload sensor readings or patch security issues, while long range technologies can provide nation wide connectivity. In order to select the best available LPWAN technology, a novel network detection algorithm was designed that provides parameter trade-offs between the network discovery time and desired reliability. The algorithm was shown to help save energy when configured correctly and provide OTAUs for a Sigfox device, which by itself is only capable of receiving 32 bytes.

Whereas Chapters 2 to 4 delivered device and network interoperability for multimodal LPWAN devices, Chapter 5 tries to come up with a solution to provide interoperability at a semantic level. The Chapter investigates how data coming from LPWANs can be prepared for the Semantic Web. Technologies that have been around for years, such as Resource Description Framework (RDF), weren't optimized for constrained networks. Therefore, a mapping tool was added to the architecture that can convert IoT (management) semantics to an RDF-based ontology. The evaluation showed that using this approach, it is possible to convert regular IoT data to Linked Data (LD) while consuming substantially less energy compared to a regular LD approach.

Both Chapter 4 and 5 showed the need of a central entity in order to provide the targeted interoperability and multimodality. In order to cope with this need, Chapter 6 presented a generic approach in developing interoperable, scalable and extensible LPWAN IoT applications. The modularity of the platform, makes it easy to integrate adapters that connect non-interoperable networks, platforms and applications. The provided platform interoperability, cross-domain interoperability and modular approach allow fast prototyping and integration of existing software. Therefore, it may in the future contribute to trans-sector innovation.

7.1 Future Work

The adoption of LPWAN technologies, and IoT technologies in general, did not reach its peak yet. Much of its potential is currently locked in vertical silos. Silos in multiple domains were addressed in this dissertation in order to move towards a horizontally interconnected LPWAN landscape. However, this dissertation only offers a few solutions to move towards scalable, interoperable, heterogeneous LPWA networks. The following section gives a non exhaustive overview of existing problems that still exist and possible avenues to investigate.

Although SCHC provides a standards-based approach for communication over LPWANs, and ideally, the developed device and context registration mechanisms could result in decentralized LPWAN systems, further research is required in order to decouple these networks from their APIs. Eventually, this could lead to a connected Wide Area landscape, were devices can roam freely between heterogeneous networks. However, just like in Mobile Internet Protocol (IP), connectivity should be achieved seamlessly, without device or user intervention. Mobile IP provides a node with the ability to retain the same IP address, while moving between networks, with the help of a Foreign Agent and a Home Agent. These concepts might be translated to SCHC-enabled LPWANs, which could lead to global LPWAN connectivity.

Currently, LPWAN deployments are not as congested as the 2.4 Gigahertz (GHz) band, where scheduling algorithms are being used for fairness among technologies that use the same frequency space. However, with the upcoming trend of IoT, this is expected to happen in the Industrial, Scientific and Medical (ISM) band too. Due to the use of sub-GHz radio frequencies, a single LPWAN base station has a large coverage area that can support a high number of connected devices (> 1000) and since most LPWAN technologies do not employ an advanced MAC protocol, scaling up the number of devices may result in a network collapse. Several solutions have been proposed for single technology deployments, such as the use of Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA), Request to Send and Clear to Send (RTS/CTS) and time based scheduling. The latter has been proven to provide the most efficient and scalable solution. However, co-located networks may still interfere with each other. Therefore, there is a need for cross-technology scheduling and cross-gateway coordination in multimodal networks.

This dissertation focused on the widespread Internet Protocol; the heart of the Internet architecture. It is a simple, universal layer that allows devices to communicate with a destination directly. However, the growth of the Internet led to a network that is mainly used to distribute content. This resulted in

many complex Internet protocols that solve distribution problems over a communications network. Therefore, Van Jacobsen introduced the concept of Content Centric Networking (CCN) in 2006 to provide an alternative to the Internet Protocol Specification (RFC791). This proposes an evolution from the current, location-based networking to content-based networking, leveraging on the idea that people value the Internet for *what* content it contains and not *where* it resides. From this project, the Named Data Networking (NDN) project emerged to investigate Jacobsen' proposal. The IoT has been identified as a potential deployment area for Information Centric Networks (ICNs) and therefore received increasing attention from the research community. However, extensive header values make it unfeasible to use NDN over networks with limited bandwidth capabilities. Further investigation should clarify whether it can be used directly on top of LPWAN networks. NDN provides native support for mobility, therefore, a comparative study might shed light on which of both approaches is the most feasible. The header overhead of NDN should be studied, as it can have a large impact on the energy consumption of constrained devices. SCHC could play a role in communication optimization in NDN networks over constrained wireless links.

The architecture that was presented in Chapter 6 has at its core a directory that contains all Thing Descriptions (TDs), based on the World Wide Web Consortium (W3C) Web of Things (WoT) specification. The directory exposes an API that can be used to create, read, update and delete a TD and to browse things. Hence, the directory can be used by other Web resources to discover the connected devices. FLINT has implemented a small part of the WoT specification for further exploration of the W3C specification for LPWAN devices. Since most LPWAN devices are sleeping end-points that can not be reached directly, digital twins can reflect their last known state. However, applications should be redirected to a device its Linked Data Platform (LDP) end-point in order to retrieve historical measurements. Furthermore, FLINT has developed a solution to deal with multimodality in constrained networks. These solutions can be used as a leverage point for Web of Things platforms.

A

Device Discovery and Context Registration in SCHC Networks

This Chapter provides additional material to clarify various proposals from chapter 3.

A.1 SCHC Neighbor Discovery

Static Context Header Compression (SCHC)-compressed Neighbor Discovery (ND) builds on optimized ND, which introduced the Address Registration Option. When using the Address Registration Option (ARO), several requirements are imposed to the client [1], which can be optimized using SCHC. First of all, the address to be registered must be the Internet Protocol Version 6 (IPv6) source address of the message. Therefore, the client should perform stateless address configuration first and send the generated IPv6 Interface Identifier (IID) address in the residue of the SCHC packet. Furthermore, the Destination Address field should be set to the unicast address of the target. This, however, can be reconstructed by the receiving gateway and can therefore be elided. Furthermore, a Source Link-Layer Address Option (SLLAO) must be included, the Extended Unique Identifiers (EUI)-64 field can be computed using SCHC and can therefore be elided. The compression rules for a Network Server (NS)/Neighbor Advertisement (NA) exchange is given in Table A.1. The table also shows that the SCHC-

compressed ARO message must not carry the target Medium Access Control (MAC) address of the SCHC gateway, as this can be reconstructed using the values of the Router Advertisement (RA). The ARO also includes the lifetime of the registration in the upward direction (in units of 60 seconds) and the status of the registration in the downward direction.

Table A.1: SCHC neighbor discovery NS/NA for Ack-on-Error and Ack-Always.

Field	**FL**	**DI**	**TV**	**MO**	**CDA**
IPv6 Version	4	BI	6	`equal`	`not-sent`
IPv6 Traffic Class	8	BI	-	`ignore`	`not-sent`
IPv6 Flow Label	20	BI	-	`ignore`	`not-sent`
IPv6 Length	16	BI	-	`ignore`	`compute-*`
IPv6 Next Header	8	BI	58	`equal`	`not-sent`
IPv6 Hop Limit	8	BI	255	`ignore`	`not-sent`
IPv6 Src Prefix	64	UP	-	`ignore`	`not-sent`
IPv6 Src IID	64	UP	-	`ignore`	`value-sent`
IPv6 Src Prefix	64	DO	-	`ignore`	`not-sent`
IPv6 Src IID	64	DO	-	`ignore`	`not-sent`
IPv6 Dst Prefix	64	UP	-	`ignore`	`not-sent`
IPv6 Dst IID	64	UP	-	`ignore`	`not-sent`
IPv6 Dst Prefix	64	DO	-	`ignore`	`not-sent`
IPv6 Dst IID	64	DO	-	`ignore`	`not-sent`
ICMPv6 NS	8	UP	135	`equal`	`not-sent`
ICMPv6 NA	8	DO	136	`equal`	`not-sent`
ICMPv6 Code	8	BI	0	`equal`	`not-sent`
ICMPv6 Checksum	16	BI	0	`ignore`	`compute-*`
ICMPv6 Reserved	32	UP	0	`ignore`	`not-sent`
ICMPv6 R	1	DO	1	`ignore`	`not-sent`
ICMPv6 S	1	DO	1	`ignore`	`not-sent`
ICMPv6 O	1	DO	0	`ignore`	`not-sent`
ICMPv6 Reserved	29	DO	0	`ignore`	`not-sent`
ICMPv6 Target Addr	128	BI	-	`ignore`	`not-sent`
ICMPv6 SLLAO	8	UP	1	`equal`	`not-sent`
ICMPv6 Length	8	UP	1	`equal`	`not-sent`
ICMPv6 EUI-64	64	UP	-	`ignore`	`compute-*`
ICMPv6 ARO	8	BI	33	`equal`	`not-sent`
ICMPv6 Length	8	BI	2	`equal`	`not-sent`
ICMPv6 Status	8	UP	0	`equal`	`not-sent`
ICMPv6 Status	8	DO	-	`ignore`	`value-sent`
ICMPv6 Reserved	24	BI	0	`ignore`	`not-sent`
ICMPv6 Lifetime	16	BI	-	`ignore`	`value-sent`
ICMPv6 EUI-64	64	BI	-	`ignore`	`not-sent`

A.2 SCHC Context Options

SCHC Context Options are divided in Fixed Size Context Options and Variable Size Context Options. Both message structures have a fixed flag field, indicating either the fixed (0) or variable (1) type. The layer field (2 bits) indicates which layer is targeted; (0) the network layer, (1) transport layer, or (2) application layer, leaving room for inclusion of other protocols. The fixed header employs a 4-bit type field, which reflects the header fields of the targeted protocol in the order of appearance. The variable header on the contrary, uses an 8-bit type field to indicate the type of, for example, a Constrained Application Protocol (CoAP) option. A CoAP message can therefore be constructed using variable and fixed sized options. The variable sized fields require the Field Position (FP) (3 bits) option, as they can appear multiple times in a single header. The Field Length indicates the Target Value (TV)'s number of bits and can be used to calculate the total length of the Internet Control Message Protocol (ICMP) for IPv6 (ICMPv6) message. Finally, both structures indicate the Direction using the DI field (2 bits) to indicate the direction (0) Up, (1) Down, or (2) Bidirectional. For fixed sized headers, the DI field and the type are used to form a unique combination inside the rule. Fields in variable sized headers can be distinguished by means of FP and Type.

Next, the Matching Operator (MO) field (2 bits) indicates which Matching Operator will be used.

- 0x00 indicates the `equal` MO
- 0x01 indicates the `ignore` MO
- 0x02 indicates the `MSB(x)` MO
- 0x03 indicates the `match-mapping` MO

Both `MSB(x)` and `match-mapping` MO, however, require extra parameters. The `MSB(x)` MO will only transmit `x` bits of the Target Value, while the `match-mapping` MO implements an array of which the index is sent as a compressed value. In order to configure this information, a SCHC Context Option with the MO field set to 0x02 or 0x03, must be succeeded by an 8-bit SCHC Compression Action Option, given in Figure A.1.

When the `MSB(x)` MO is targeted, the Parameter Value field carries the number of bits that must be transmitted. When configuring the `match-mapping` MO, the Parameter Value is used to indicate the length of the array. In order to distinguish between different entries, the Target Value of any array must be constructed using Concise Binary Object Representation (CBOR).

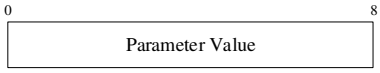


Figure A.1: SCHC compression action option.

Finally, the Compression Decompression Action (CDA) field (3 bits) is used to designate which Compression/Decompression Action to use

- 0x00 `not-sent`
- 0x01 `value-sent`
- 0x02 `mapping-sent`
- 0x03 `LSB`
- 0x04 `compute-*`
- 0x05 `DevIID`
- 0x06 `AppIID`

A.3 SCHC Parameter Option

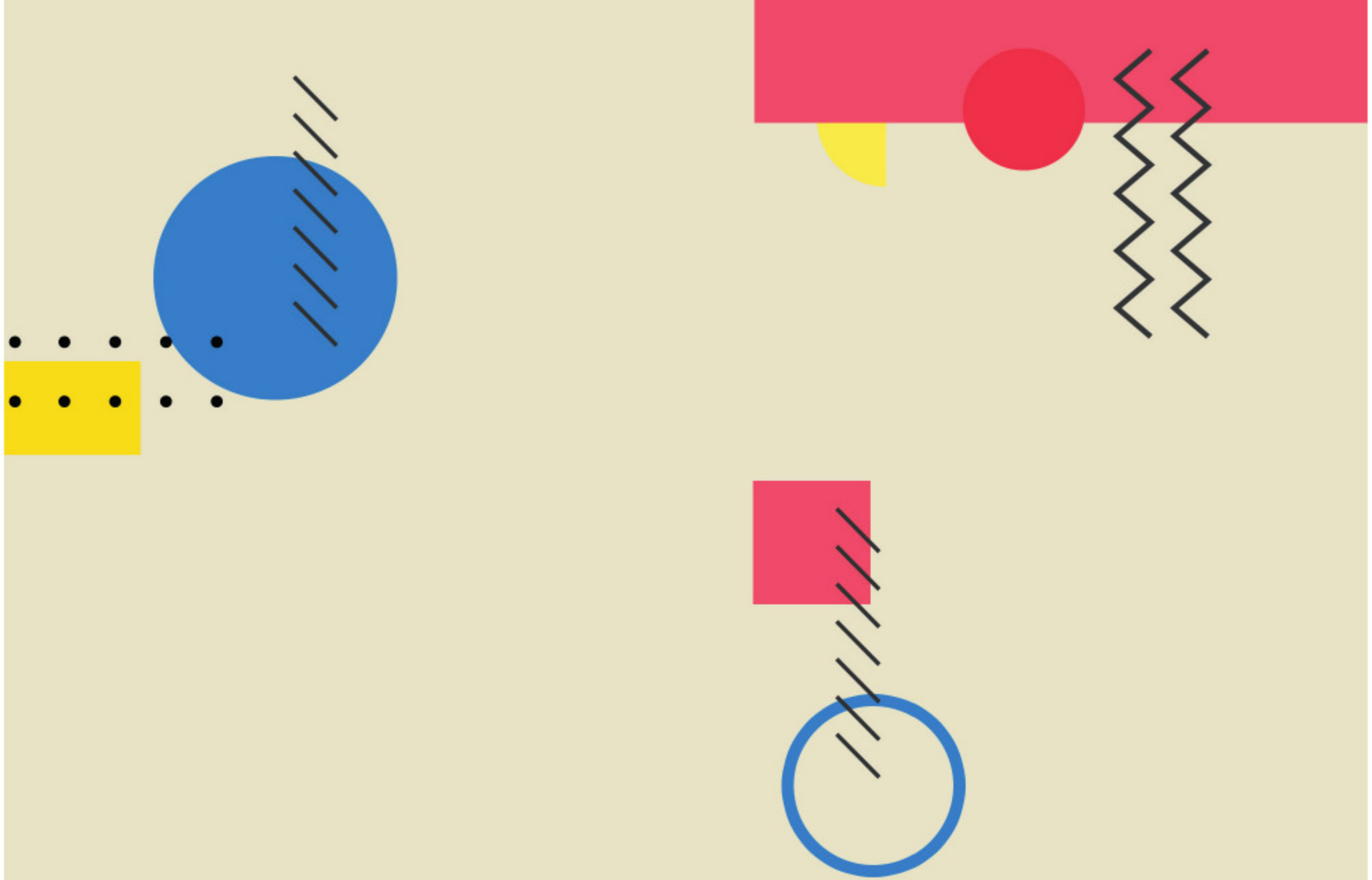
The SCHC Parameter Option (SPO) can set the following values:

- Type: 41
- `RULE_ID_SIZE`: 4 bits indicating the default number of bits for a rule id
- `WINDOW_SIZE (M)`: 6 bits indicating the default window size in bits. If set to 0, no windows are used.
- `MAX_ACK_REQ`: 4 bits indicating the default maximum allowed acknowledgment requests
- `RCS_SIZE`: 6 bits to indicate the default size used to calculate the Cyclic Redundancy Check (and default polynomial `0xEDB88320`, with size equal to `RCS_SIZE`)
- `DTAG (T)`: 3 bits indicating the Datagram Tag size. If set to 0, no more than 1 SCHC packet can be in transit for each fragmentation rule id.
- `P`: the value of the padding bits

- `RETRANSMISSION_TIMER`: 16 bits in units of 60 seconds, resulting in a maximum of 45 days and 12 hours for reliability modes to time out while waiting for an acknowledgment
- `INACTIVITY_TIMER`: 16 bits in units of 60 seconds representing the time before a receiver will abort waiting for a SCHC message

References

[1] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs).* Technical Report RFC6775, RFC Editor, November 2012. Available from: https://www.rfc-editor.org/info/rfc6775, doi:10.17487/rfc6775.



An abstract representation of interoperability: unifying dissimilar systems.