information

MDPI

Article

Device Discovery and Context Registration in Static Context Header Compression Networks

Bart Moons *,†, **Eli De Poorter** † **and Jeroen Hoebeke** †

IDLab, Department of Applied Engineering, University of Ghent—imec, 9056 Gent, Belgium; eli.depoorter@ugent.be (E.D.P.); jeroen.hoebeke@ugent.be (J.H.)

* Correspondence: bamoons.moons@ugent.be

† Current address: Technologiepark-Zwijnaarde 126, 9052 Ghent, Belgium.

Abstract: Due to the limited bandwidth of Low-Power Wide-Area Networks (LPWAN), 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 Static Context Header Compression (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 Lightweight Machine to Machine (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 (LoRa) devices still have a 80% chance to successfully complete the registration flow in a network with a 50% Packet Error Ratio. Briefly, the work presented in this paper delivers bootstrapping tools to constrained, SCHC-enabled networks while still being able to reduce energy consumption.

Keywords: LPWAN; Internet of Things; Static Context Header Compression; IPv6; standardization

Citation: Moons, B.; De Poorter, E.; Hoebeke, J. Device Discovery and Context Registration in Static Context Header Compression Networks. *Information* **2021**, *12*, 83. https://doi.org/10.3390/info12020083

Academic Editor: Ruggero Lanotte

Received: 15 December 2020
Accepted: 12 February 2021
Published: 16 February 2021

1. Introduction

The success of the internet introduced the TCP/IP (Transmission Control Protocol/Internet Protocol) 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 UDP/IP (User Datagram Protocol/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 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 (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 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) working group [2].

However, the lack of bandwidth in long range communication technologies started the formation of the LPWAN working group (WG), as there is often no room for the 6LoWPAN protocol overhead [3]. The effort of this WG resulted in the Static Context Header Compression (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 (Cyber Security for Consumer Internet of Things), 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 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 2 describes the Generic Framework for Static Context Header Compression. Section 3 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 4. Possible solutions build upon this and are presented in Sections 5 and 6. Finally, both solutions are evaluated in Section 7 and the advantages and disadvantages for both solutions are discussed in Section 8.

2. Static Context Header Compression

As explained in the previous Section, sensor devices can implement the Static Context Header Compression adaptation layer protocol for standardized internet connectivity. This protocol, defined in RFC 8724, resides between the Medium Access (MAC) layer and the IPv6 layer and is an attempt to bring IPv6 connectivity to constrained embedded devices that communicate over an often even more constrained wireless link [5].

2.1. SCHC Framework

The SCHC framework leverages on the idea that the task of these sensor devices mainly consists of data delivery and consequently, an application on top of the sensor device remains more or less the same for a relatively long period of time. For example, an air quality sensor will deliver data to the same broker or data service and only requires very sporadic downlink communication. The information about the header fields does not change much over time, and therefore remains static. This information is stored in a context known to both sides of the network and only a small context identifier is often sufficient to represent the full network layer, transport layer and application layer headers. RFC 8724 also defines a reliable fragmentation and reassembly mechanism, which can be used to support the IPv6 Maximum Transfer Unit (MTU). Fragmentation will occur when the compressed headers and payload still exceed the underlying link layer. The fragmentation layer becomes increasingly important when distributing large files over constrained links, such as over-the-air firmware updates. In this standard, three reliability modes are defined:

1. `no-ack` does not define any reliability other than what is provided by the link layer.
2. `ack-on-error` acknowledges every erroneous window.

3. `ack-always` acknowledges every window.

2.2. SCHC Context

In order to compress or fragment a packet, SCHC relies on a static context known to both sides of the network. The context consists of one or more rules that are distinguished by means of a unique identifier. 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 in the rule.

- `equal` looks for an exact match
- `ignore` ignores the field
- `MSB(x)` compares the first `x` bits
- `match-mapping` compares a list of entries

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

The rule id and possibly compressed values (called residue) are sent to the other side of the network. The decompressor will use the Target Value and the residue to reconstruct the original value.

2.3. Related Work

Static Context Header Compression started receiving increasing attention from the research community. In both [3,6], 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 [7]. Both authors conclude that a novel adaptation standard is needed, as 6LoWPAN does not provide enough flexibility for LPWAN communication technologies. The authors of [6] continue their work in [8], where a central Administration Management Server (AMS) manages the context in order to support roaming of devices between different Long-Range Wide-Area Network (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. [9] 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 Dynamic Name Resolution (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 [10]. The authors proposed a novel matching operator, i.e., the dummy-mapping 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. 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, LoRa, Weightless, NB-Fi, NB-IoT, eMTC, EC-GSM, and DASH-7, among others [11]. 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 Application Programming Interfaces (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 [12]. However, due to the efforts of the 6LoWPAN working group, middle-boxes of service providers could be decoupled, which led to easier deployment of applications and emergence of new businesses.

Figure 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 2. This, however requires a bootstrapping protocol in order to perform context and device discovery.

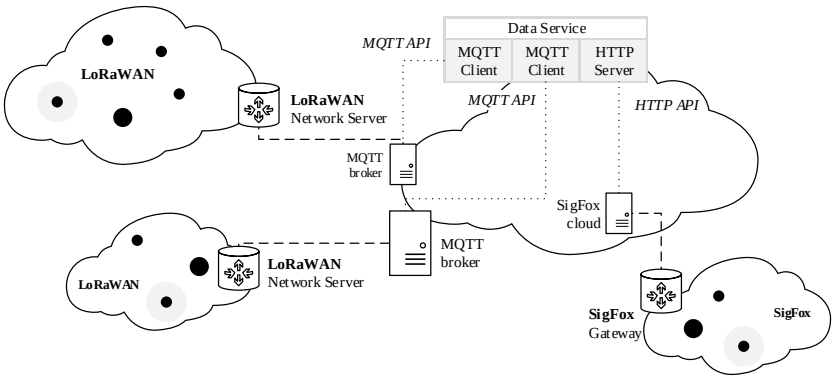


Figure 1. The current Low-Power Wide-Area Network landscape: centralized components deliver data to translation units.

Second, 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 [13]. This way, LoRaWAN devices could securely roam between different LoRaWAN networks. In the architecture of Figure 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 current SCHC standard does not allow such dynamic configuration, which would result in a per-packet header overhead, due to (partially) uncompressed headers.

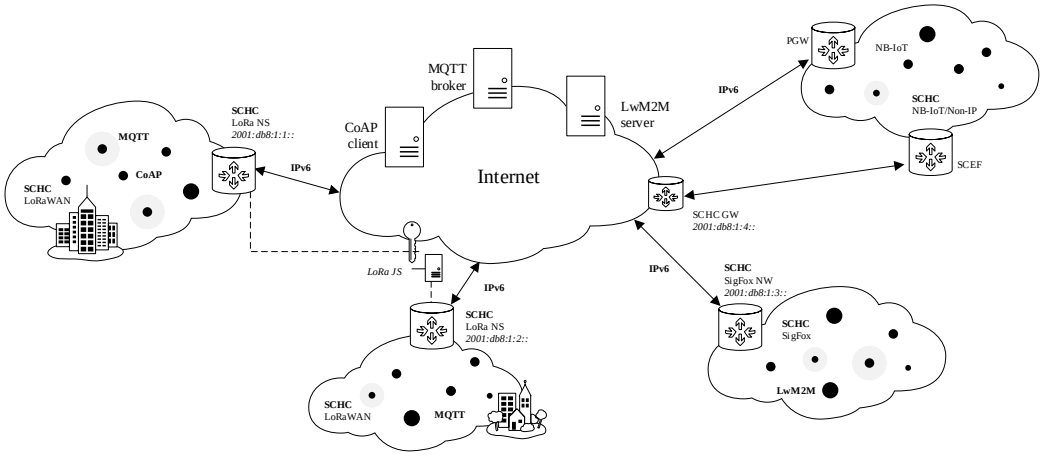


Figure 2. End-to-end secured, decentralized IPv6 Low-Power Wide-Area Network.

Third, 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. Lightweight Machine to Machine (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 State-of-The-Art (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.

4. 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 Neighbor Discovery protocol (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 Low-Power Wireless Personal Area Networks (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 (NETCONF) protocol, which can be used to monitor networks and manage devices and their interfaces.

4.1. 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 Neighbor Discovery (ND) specification [14]. 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 Neighbor Discovery, 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 Neighbor Solicitation 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 Low-Power Wide-Area Networks. 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 [15].

4.2. Optimized Neighbor Discovery Protocol

The basic IPv6 Neighbor Discovery mechanism has been optimized for 6LoWPAN networks, which resulted in 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 [16], 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 [17]. The Neighbor Registration and Periodic Updates section of Figure 3 give an overview of the optimized ND protocol.

Optimized ND inherits all functions defined in the ND specification, however, introduced several optimizations in order to cope with the low power nature of LoWPANs.

4.2.1. 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 an 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.

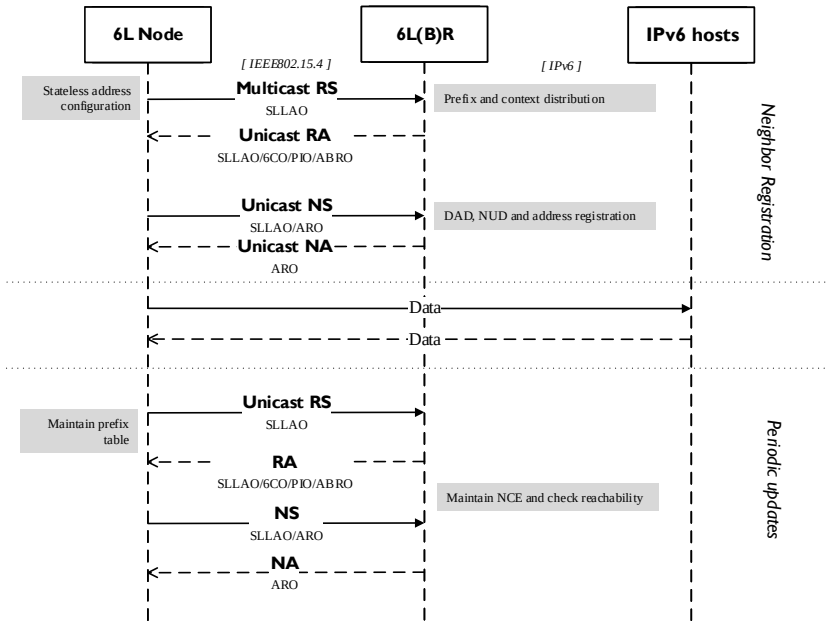


Figure 3. 6LoWPAN optimized neighbor discovery.

4.2.2. 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.

4.2.3. 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.

4.3. 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 [18]. 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 Calls (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, and Delete (CRUD)), used to access and update the datastores of devices connected to the network. As NETCONF runs over a Secure Shell Session (SSH), the IETF developed RESTCONF to access datastores and manage network equipment over HTTP by meanst of web applications. The RESTful POST, PUT, GET, and DELETE methods can be mapped easily to NETCONF's CRUD operations.

CORECONF

Similarly, the IETF started the development of the CoAP Management Interface (CORECONF), mapping the various CRUD operations of NETCONF to the REST methods available in the Constrained Application Protocol (CoAP) to manage constrained devices [19]. 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 [20].

4.4. LwM2M

Another IoT management protocol has been developed by the Open Mobile Alliance (OMA) and goes under the name of LightWeight Machine-to-Machine (LwM2M) [21]. 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.

4.5. Conclusions

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.

5. 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 [16]. 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).

5.1. 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 [22]. 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.

5.2. 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.

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.

Registration

A sensor device can send a registration message by setting the source IPv6 address to an unspecified one. As Table 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 6.

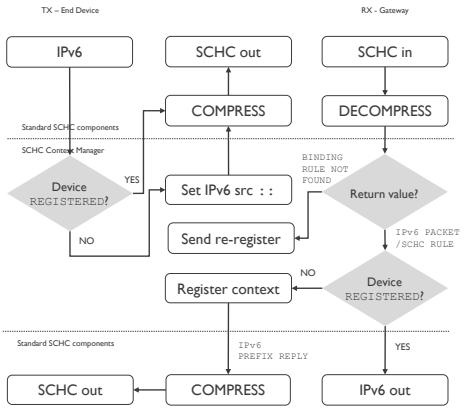
Table 1. Static Context Header Compression (SCHC) registration request.

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-*`

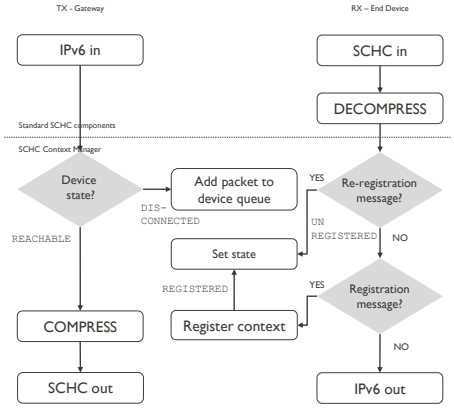
As depicted in Figure 4a, every SCHC packet arriving at the Network Gateway, is decompressed first. The return value of the decompressor is then passed to the SCM, where either

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; and
3. or 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.



(**a**) SCM up



(**b**) SCM down

Figure 4. FSMfor the Context Manager in both directions.

Figure 4b shows the FSMfor traffic flowing in the downward direction. If a device, such as a Sigfox or LoRa 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.

5.3. 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 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 RA. The RA contains a Source Link Layer Address Option (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 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.

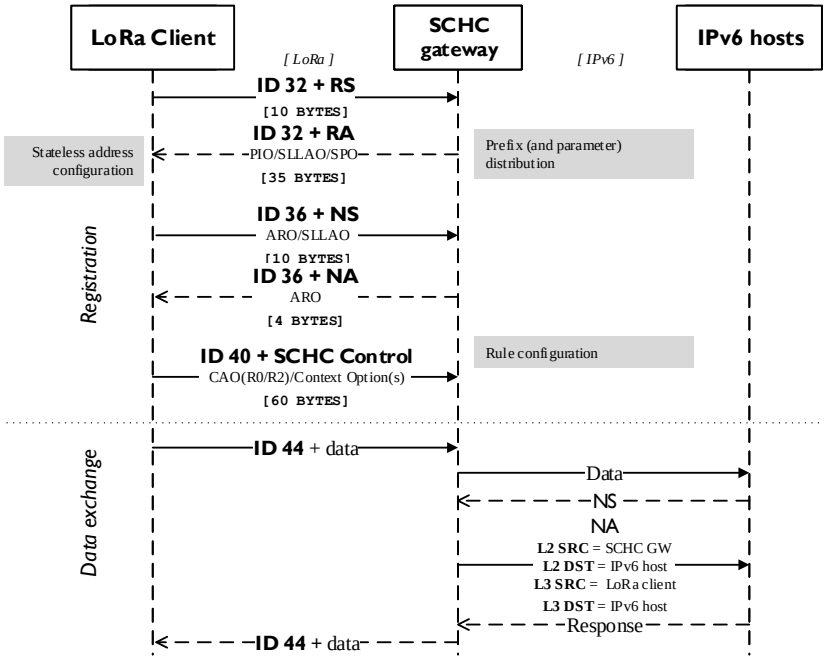


Figure 5. ICMPv6-based SCHC device and context registration.

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.

6. 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.

6.1. 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.

6.1.1. 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 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.

6.1.2. 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.

6.1.3. 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 LoRa Class A device performing the different steps in the SCHC registration process is given in Figure 6.

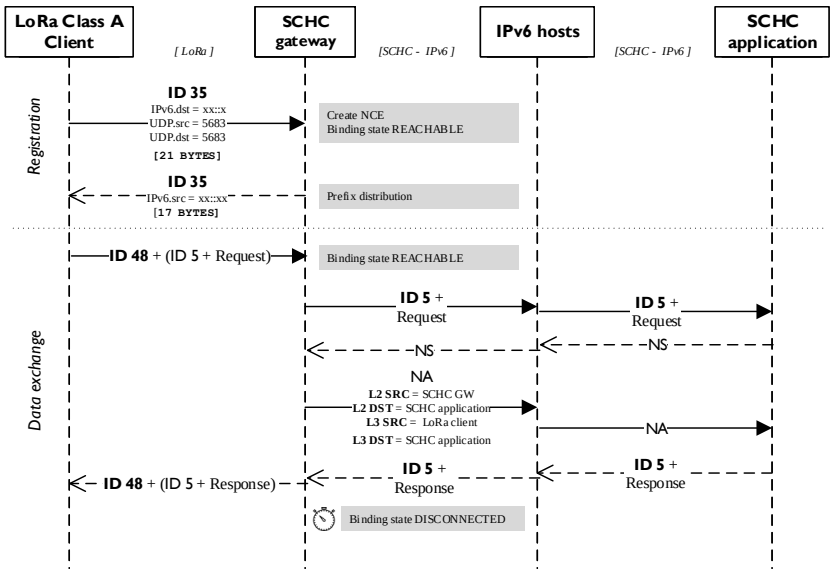


Figure 6. SCHC-based device and context registration.

6.1.4. 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., this cannot be expanded towards the Transmission Control Protocol (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.

6.2. 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 7 illustrates the structure of the SCHC Control Message.

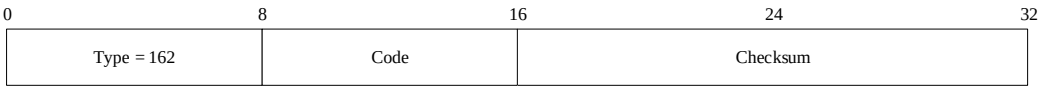


Figure 7. ICMPv6 header for SCHC control messages.

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

6.2.1. Context Advertisement Object and Context Option

In order to indicate which rule is targeted, the Context Advertisement Object (CAO), illustrated in Figure 8, can be attached to a SCHC Control Message.

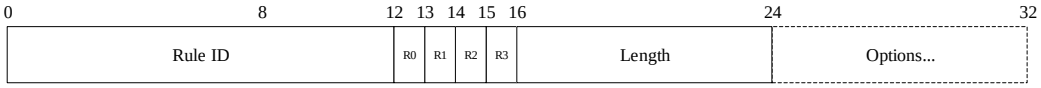


Figure 8. SCHC context advertisement object.

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 can be up to 256. This is required to know when Fragmentation Context Options will follow, which will be introduced in Section 6.2.3. 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 1.

Listing 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 [23]. 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 9 and 10.

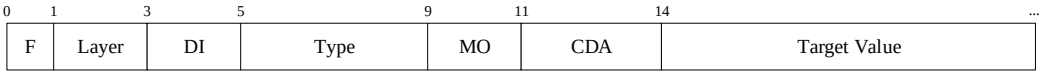


Figure 9. SCHC fixed size context option.

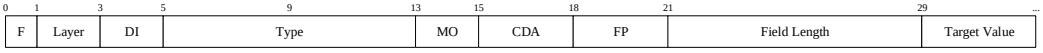


Figure 10. SCHC variable size context option.

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. 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 B.

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.

6.2.2. 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.

6.2.3. 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 11. The different fields carry several parameters that must be defined in a SCHC profile. A more detailed explanation of the different fields is given in Appendix C.

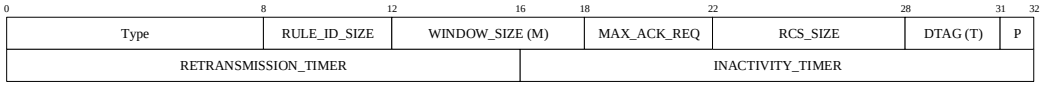


Figure 11. ICMPv6 SCHC parameter option.

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 12.

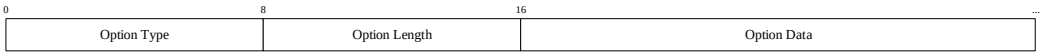


Figure 12. ICMPv6 SCHC fragmentation context option.

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.

7. 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 FCN of 3 bits. No DTAG was used and a Cyclic Redundancy Check (CRC) of 32 bits was used during fragmentation sequences.

7.1. Comparison

To start with, a comparison of both solutions is given in Table 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 evolutions of networks. This control, however, comes with a larger transmission and energy overhead, which will be shown in the next Section.

Table 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

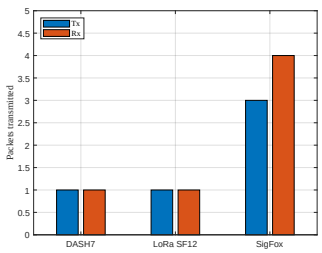
7.2. Registration Overhead

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. 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 Reassembly Check Sequence (RCS), which was set to 32 bits. Both mechanisms, however, still fit the MTU of a LoRa SF12 device, without the need for fragmentation.

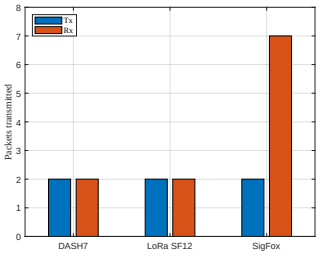
Depending on the request/response sequence and possible fragmentation, a higher number of packets is exchanged. This is illustrated in Figure 13. SCHC Registration only requires two packets (1 up and 1 down) for devices with a higher order bandwidth, such as LoRa and DASH7. Sigfox devices, on the contrary, 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 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.

Table 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.

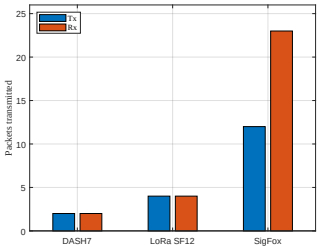
	Mode	Direction	Overhead (bytes)
Registration	Non-fragmented	Up	21
		Down	17
	Ack-on-Error	Up	28
		Down	25
Extended Reg.	Non-fragmented	Up	26
		Down	17
	Ack-on-Error	Up	34
		Down	25
SCHC RA/RS with PIO/SLLAO/SPO	Non-fragmented	RS	10
		RA	35
	Ack-on-Error	RS	14
		RA	44
SCHC NA/NS with ARO/SLLAO	Non-fragmented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



(**a**) SCHC Registration



(**b**) SCHC Neighbor Discovery



(**c**) Optimized Neighbor Discovery

Figure 13. The total packet overhead for a single registration attempt using both registration mechanisms for different technologies.

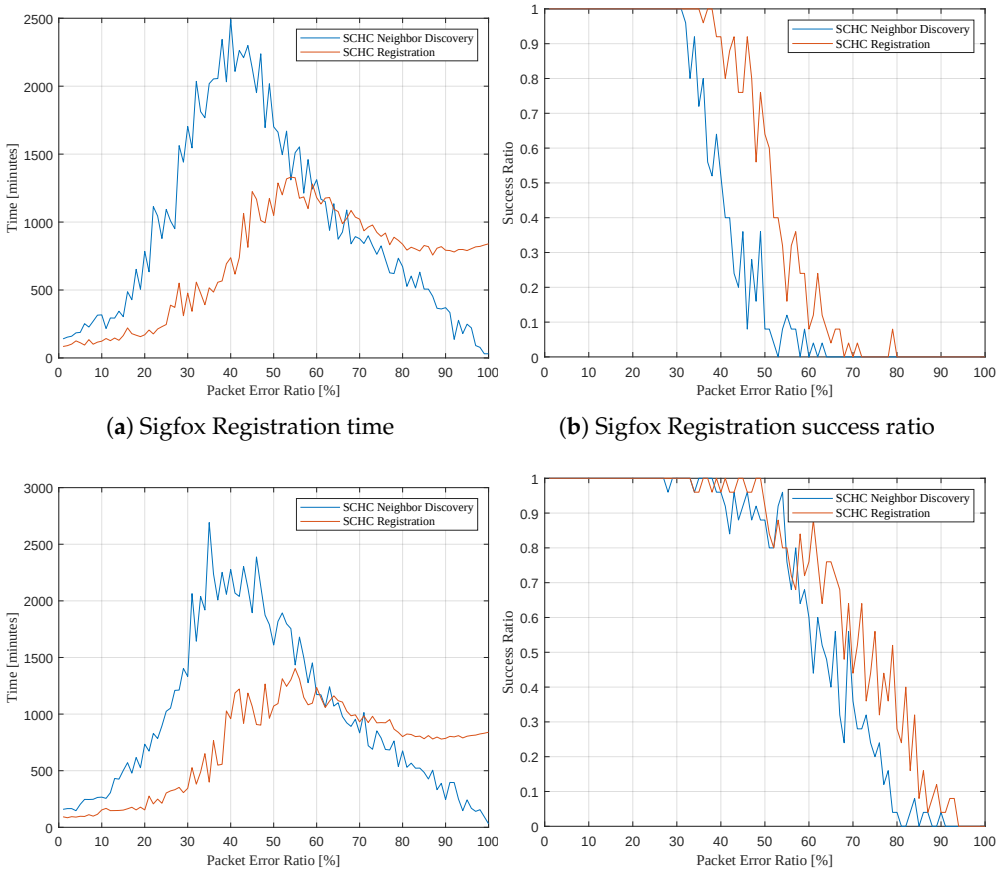
Registration Time

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 packet error ratio (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 [24].

Highly dense networks have 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 LoRa SF12 device and a Sigfox device are given in Figure 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 LoRa 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 registration time. The graph shows that at least 92 min are required to register a Sigfox device to the network using SCHC Registration and 160 min are required for SCHC ND in a best case scenario. However, only 4 to 7 min are required over a LoRa network. Every registration attempt incorporates the time on air and a 1% duty cycle restriction.



(**a**) Sigfox Registration time

(**b**) Sigfox Registration success ratio

(**c**) LoRa SF12 Registration time

(**d**) LoRa SF12 Registration success ratio

Figure 14. Registration time for different packet error ratios (PERs).

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 LoRa 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 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

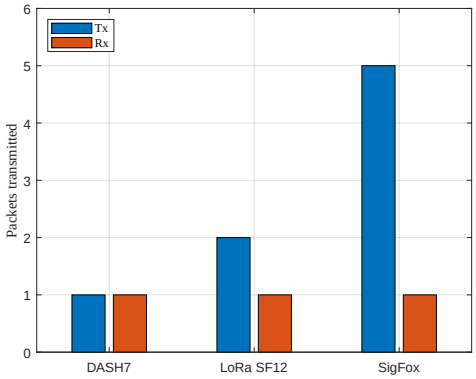
RA in response to a RS (which has a very low binary overhead) and consequently will not start the NS/NA sequence. This way, the RS can act as a PER indicator and can be used as such to increase the back-off period. In a fragmented environment, the SCHC Registration time decreases after the point at which 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.

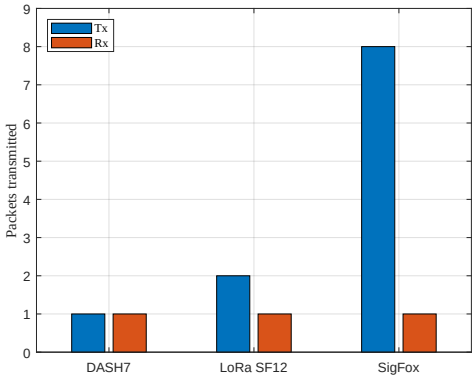
7.3. SCHC ND: Context Configuration Overhead

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 15a. A single acknowledgment suffices to acknowledge the fragmentation window in case no packet was lost.



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



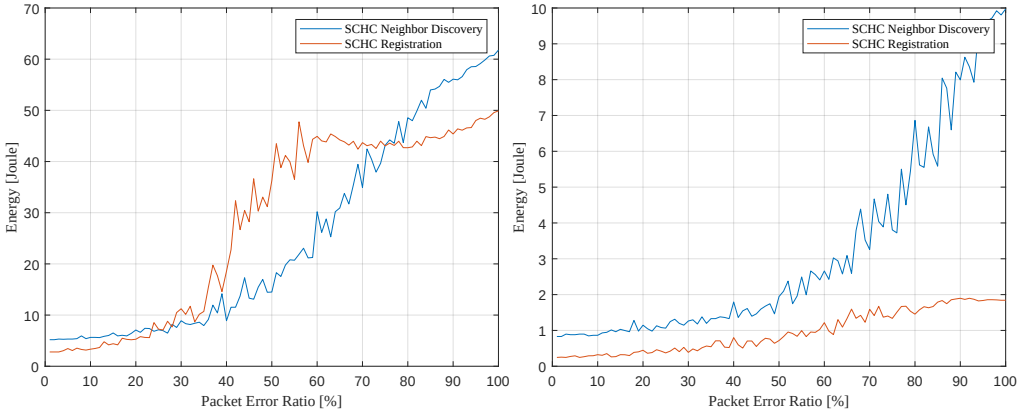
(**b**) SCHC complete IPv6/UDP rule context configuration overhead in `ack-on-error` mode

Figure 15. SCHC IPv6/UDP context configuration.

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 15b.

Energy Overhead

In exchange for the provided flexibility using the context configuration protocols, a higher energy consumption can be expected. We take the PER to measure the extra energy consumption in the situation described in Figure 15a, compared to a static context. The results are presented in Figure 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.



(**a**) Sigfox context registration overhead (**b**) LoRa context registration overhead

Figure 16. Energy overhead in order to configure the SCHC context for different PERs.

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.

7.4. 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 17. A client is configured with the credentials of the LwM2M bootstrap server, to which it can request the information of the LwM2M server.

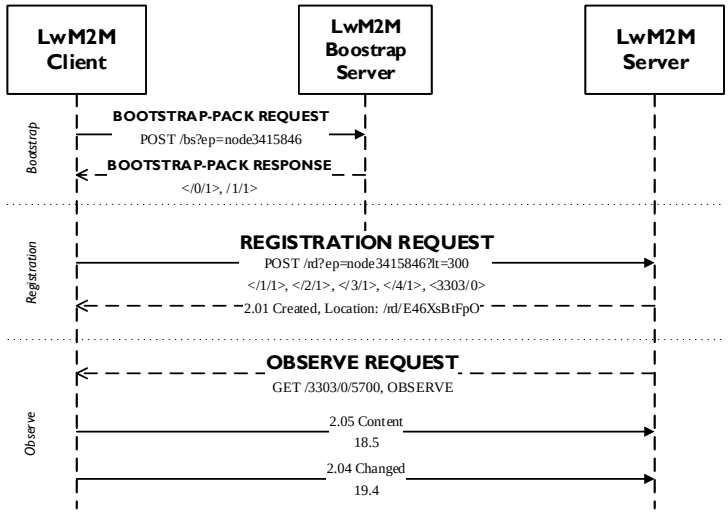
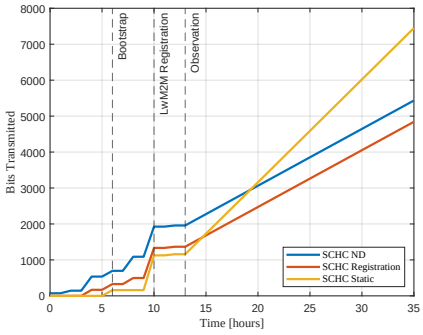


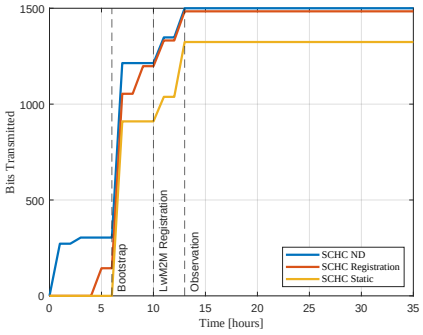
Figure 17. LwM2M bootstrap, registration, and observe request.

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 18 shows the cumulative bit overhead for 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.

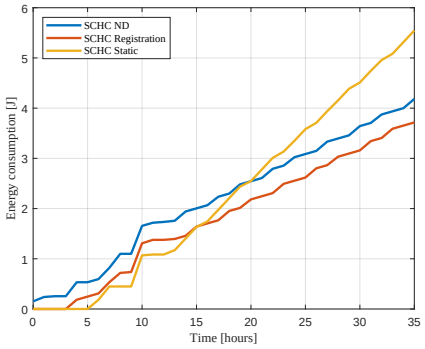
The fifth transmission initiates the LwM2M bootstrap-pack sequence. Until this point, the static configuration clearly outperforms both solutions. After receiving the LwM2M server objects (the 6th transmission in downlink, which explains the large spike), both registration mechanisms again register their 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.



(**a**) Cumulative bit overhead in uplink



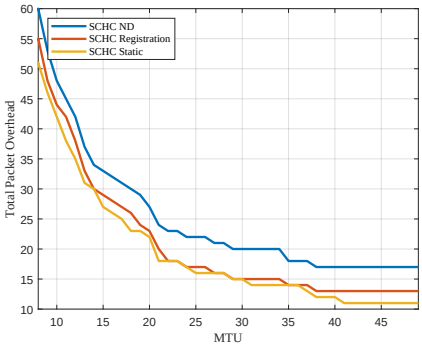
(**b**) Cumulative bit overhead in downlink



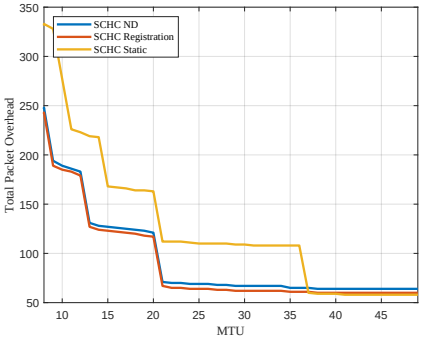
(**c**) Cumulative energy consumption

Figure 18. The cumulative bit and energy overhead for the static and dynamic solutions for a LoRa SF12 device observing its temperature value every hour.

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 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 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.



(**a**) Total packet overhead per MTU for 1 notification

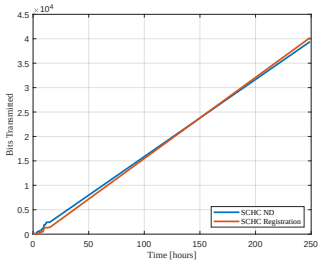


(**b**) Total packet overhead per MTU for 48 notifications

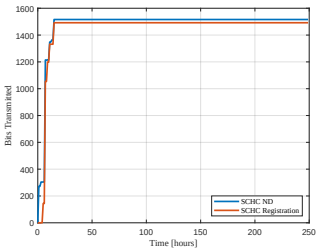
Figure 19. The total packet overhead for the complete LwM2M registration flow and different number of observation notifications.

7.5. CoAP Compression

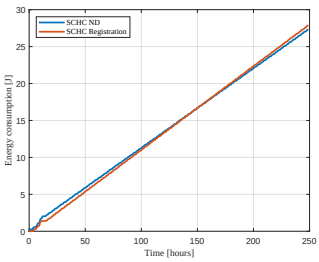
The previous sections showed the significance of both registration mechanisms compared to the current static solution. However, as the SCHC Registration mechanism requires double compression, this will also impact the total bit overhead. In this evaluation, we suppose that a CoAP context is deployed 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 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.



(**a**) Cumulative bit overhead in uplink



(**b**) Cumulative bit overhead in downlink



(**c**) Cumulative energy consumption

Figure 20. The cumulative bit and energy overhead for both dynamic solutions for a LoRa SF12 device.

8. 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 versa. This is something that is not supported by the current DNS-based solution from Bernard et al. [9].

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 end-points, 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.

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 Working Group, might be used in the future to provide interoperability among SCHC contexts. Once Schema Item Identifiers (SID) 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.

9. Future Work

9.1. 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" down-

link 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.

9.2. 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 [25], 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 [26]. 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.

9.3. Security

As the SCHC Control Messages contain sensitive information, security should be looked at into more detail. This can be done similarly to RPL, where the higher order bit of the Code field (0x80) can be used to denote whether the message has security enabled [27]. Next, a security field should be added between the base object and the ICMPv6 header to support confidentiality and integrity.

Furthermore, the SCHC registration mechanism requires a more in-depth study to cope with security issues.

9.4. Information Centric Networking

Van Jacobsen introduced the concept of Content-Centric Networking (CCN) in 2006, which 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 [28]. 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.

A convergence layer, called ICNLoWPAN, has already been proposed to support ICN over LoWPANs and benefits from the compression and fragmentation tools available in 6LoWPAN [29]. Similarly, SCHC could be used to bring NDN to LPWAN networks. The SCHC framework could compress the NDN header URI, composed out of TLV fields, similar to CoAP URI paths. Furthermore, the fragmentation mechanisms could be reused in order to support large NDN messages that do not fit the small MTU sizes of LPWAN technologies.

How this context is distributed requires a more in-depth study. The solutions proposed in this work place the initiative to distribute the compression context with the sensor device. In an ICN, however, devices only start transmitting data when there is an interest for it. The heavy use of multicast in NDN networks to inform nodes about an interest will require an adaptation layer at the network gateway. Similar to ICNLoWPAN, the Pending Interest Table (PIT) of the Network Gateway and the end-devices could be extended in order to keep track of the compression context of the next hop.

9.5. Mobility

Finally, this paper presents solutions which can be used for LPWAN devices to connect to foreign networks. However, devices moving around in the IPv6 internet will not be able to maintain higher-layer connections when they change their location [30]. Mobility support is therefore particularly important for roaming devices and requires a more in depth study to cope with LPWAN networks. As this is something that is natively supported by NDN, a comparative study might shed light on which of both approaches is the most feasible.

Author Contributions: This work has been part of the PortForward project, where all the above authors have contributed to the outcome. B.M. started the investigation, conceptualization, performed the formal analysis and wrote the original draft. J.H. and E.D.P. supervised the conceptualization with core inputs for the design and revising the draft of the paper until finalization. All authors have read and agreed to the published version of the manuscript.

Funding: Part of this research has received funding from the European Union's Horizon 2020 research and innovation program under grant number 769267 (PortForward project).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

SCHC-compressed Neighbor Discovery builds on optimized ND, which introduced the Address Registration Option. When using the ARO, several requirements are imposed to the client [15], which can be optimized using SCHC. First of all, the address to be registered must be the IPv6 source address of the message. Therefore, the client should perform stateless address configuration first and send the generated IPv6 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 SLLAO must be included, the EUI-64 field can be computed using SCHC and can therefore be elided. The compression rules for a NS/NA exchange is given in Table A1. The table also shows that the SCHC-compressed ARO message must not carry the target MAC address of the SCHC gateway, as this can be reconstructed using the values of the RA. The ARO also includes the lifetime of the registration in the upward direction (in units of 60 s) and the status of the registration in the downward direction.

Table A1. 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`

Table A1. *Cont.*

Field	FL	DI	TV	MO	CDA
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`

Appendix B

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 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's (TV) number of bits and can be used to calculate the total length of the 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 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 A1.

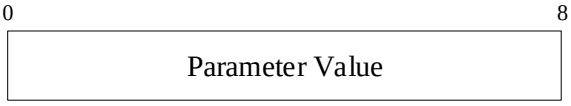


Figure A1. SCHC compression action option.

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).

Finally, the 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`

Appendix C

The 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 s, 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 s representing the time before a receiver will abort waiting for a SCHC message

References

1. Deering, S.; Hinden, R. Internet Protocol, Version 6 (IPv6) Specification. 1998. Available online: https://www.hjp.at/doc/rfc/rfc2460.html (accessed on 14 February 2021).
2. Montenegro, G.; Kushalnagar, N.; Hui, J.; Culler, D. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. 2007. Available online: https://www.hjp.at/doc/rfc/rfc4944.html (accessed on 14 February 2021).
3. Moons, B.; Karaagac, A.; Haxhibeqiri, J.; De Poorter, E.; Hoebeke, J. Using SCHC for an optimized protocol stack in multimodal LPWAN solutions. In Proceedings of the IEEE World Forum on Internet of Things (WF-IoT2019), Limerick, Ireland, 15–18 April 2019; pp. 1–6.
4. ETSI. *Cyber Security for Consumer Internet of Things*; ETSI: Sophia Antipolis, France, 2019.
5. Minaburo, A.; Toutain, L.; Gomez, C.; Barthel, D.; Zúñiga, J.C. *SCHC: Generic Framework for Static Context Header Compression and Fragmentation*; Technical Report RFC8724; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2020. [CrossRef]
6. Ayoub, W.; Nouvel, F.; Hmede, S.; Samhat, A.E.; Mroue, M.; Prevotet, J.C. Implementation of SCHC in NS-3 and Comparison with 6LoWPAN. In Proceedings of the 2019 26th International Conference on Telecommunications (ICT), Hanoi, Vietnam, 8–10 April 2019; pp. 432–436. [CrossRef]
7. Moons, B. libschc: A C Implementation of the Static Context Header Compression. 2021. Available online: https://github.com/imec-idlab/libsch (accessed on 15 February 2021).

8. Ayoub, W.; Mroue, M.; Samhat, A.E.; Nouvel, F.; Prévotet, J.C. SCHC-Based Solution for Roaming in LoRaWAN. In *Advances on Broad-Band Wireless Computing, Communication and Applications*; Barolli, L., Hellinckx, P., Enokido, T., Eds.; Springer: Cham, Switzerland, 2020; Volume 97, pp. 162–172. [CrossRef]
9. Bernard, A.; Balakrichenan, S.; Marot, M.; Ampeau, B. DNS-based dynamic context resolution for SCHC. In Proceedings of the 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
10. Abdelfadeel, K.Q.; Cionca, V.; Pesch, D. Dynamic Context for Static Context Header compression in LPWANs. In Proceedings of the 2018 14th International Conference on Distributed Computing in Sensor Systems (DCOSS), New York, NY, USA, 18–20 June 2018; pp. 35–42. [CrossRef]
11. Raza, U.; Kulkarni, P.; Sooriyabandara, M. Low Power Wide Area Networks: An Overview. *arXiv* **2016**, arXiv:1606.07360.
12. Laboratories, S. The Evolution of Wireless Sensor Networks. 2013. Available online: https://www.silabs.com/documents/public/white-papers/evolution-of-wireless-sensor-networks.pdf (accessed on 15 February 2021).
13. The Things Industries Launches Global Join Server with a Series of Device Makers to Simplify LoRaWAN Device Provisioning. 2020. Available online: https://thethingsindustries.pr.co/185845-the-things-industries-launches-global-join-server-with-a-series-of-device-makers-to-simplify-lorawan-device-provisioning (accessed on 15 February 2021).
14. Narten, T.; Nordmark, E.; Simpson, W.; Soliman, H. Neighbor Discovery for IP Version 6 (IPv6). 2007. Available online: https://www.hjp.at/doc/rfc/rfc4861.html (accessed on 14 February 2021).
15. Shelby, Z.; Chakrabarti, S.; Nordmark, E.; Bormann, C. *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*; Technical Report RFC6775; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2012. [CrossRef]
16. Watteyne, T.; Thubert, P. Efficient 6LoWPAN Neighbor Discovery applied to Multilink IoT subnets. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; pp. 642–647. [CrossRef]
17. Seliem, M.A.; Elsayed, K.M.; Khattab, A. Optimized neighbor discovery for 6LoWPANs: Implementation and performance evaluation. *Comput. Commun.* **2017**, *112*, 73–92. [CrossRef]
18. Sinche, S.; Raposo, D.; Armando, N.; Rodrigues, A.; Boavida, F.; Pereira, V.; Silva, J.S. A Survey of IoT Management Protocols and Frameworks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1168–1190. [CrossRef]
19. Veillette, M.; van der Stok, P.; Pelov, A.; Bierman, A.; Petrov, I. CoAP Management Interface (CORECONF). Internet-Draft, Internet Engineering Task Force. 2021. Available Online: https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-11 (accessed on 15 February 2021).
20. Minaburo, A.; Toutain, L. Data Model for Static Context Header Compression (SCHC). Internet-Draft, Internet Engineering Task Force. 2021. Available Online: https://datatracker.ietf.org/doc/html/draft-ietf-lpwan-schc-yang-data-model-04 (accessed on 15 February 2021).
21. OMA. Lightweight Machine to Machine Technical Specification: Core. 2020. Available Online: http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf (accessed on 15 February 2012).
22. Zuniga, J.; Gomez, C.; Toutain, L. SCHC over Sigfox LPWAN. Internet-Draft, Internet Engineering Task Force. 2019. Available Online: https://datatracker.ietf.org/doc/html/draft-zuniga-lpwan-schc-over-sigfox-06 (accessed on 15 February 2021).
23. Minaburo, A.; Toutain, L.; Andreasen, R. LPWAN Static Context Header Compression (SCHC) for CoAP. Internet-Draft, Internet Engineering Task Force. 2021. Available Online: https://datatracker.ietf.org/doc/html/draft-ietf-lpwan-coap-static-context-hc-18 (accessed on 15 February 2021).
24. Reynders, B.; Wang, Q.; Pollin, S. A LoRaWAN module for ns-3: Implementation and evaluation. In Proceedings of the 10th Workshop on ns-3 (WNS3 '18), Surathkal, India, 3 June 2018; pp. 61–68. [CrossRef]
25. Conta, A.; Deering, S.; Gupta, M. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. 2006. Available online: https://www.hjp.at/doc/rfc/rfc4443.html (accessed on 14 February 2021).
26. Toutain, L.; Dujovne, D.; Barthel, D.; Zúñiga, J.C.; Kandasamy, A. OAM for LPWAN Using Static Context Header Compression (SCHC). Internet-Draft, Internet Engineering Task Force. 2020. Available Online: https://datatracker.ietf.org/doc/html/draft-barthel-lpwan-oam-schc-02 (accessed on 15 February 2021).
27. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.P.; Alexander, R. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*; Technical Report RFC6550; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2012. [CrossRef]
28. Jacobsen, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F. Networking Named Content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT'09), Rome, Italy, 1–4 December 2009.
29. Gündoğan, C.; Kietzmann, P.; Schmidt, T.C.; Wählisch, M. ICNLoWPAN—Named-Data Networking for Low Power IoT Networks. *arXiv* **2018**, arXiv:1812.07025.
30. Perkins, C.E.; Johnson, D.B.; Arkko, J. *RFC 6275—Mobility Support in IPv6*; Technical Report RFC6275; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2011; p. 169. [CrossRef]