THE THESIS TITLE

by

Anas

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
Month Year

optional, will fill later

Table of Contents

Abstract

Static Context Header Compression (SCHC) is an adaptation layer capable of achieving large compression ratios on upper layer protocol headers (ie: IPv6, CoAP) by exploiting the presistent and predictable nature of IoT networks to make use of predefined static compression rules that act as a blueprints for the expected network traffic in which a sender can avoid transmitting the entire packet when a matching blueprint is present and instead sends a pointer to the rule. The afformentioned properities that SCHC builds on however makes it unideal for use in mobile networks in which the metadata we want to compress is variable. We look into the prospects of introducing dynamic updating of rules by evaluating the performance of scoring/weight assignment heuristics on network traffic to predict rules with improved packet coverage. Our results show that in the presence of reasonable assumptions an overall improvement on the average header size is possible.[1]

Acknowledgements

Thanks to all the little people who make me look tall.

Chapter 1

Introduction

The recent boom of IoT based networks facilitated the need for a shared networking layer to enable seemless communication between various types of devices regardless of their underlying communication medium and protocols. As such IPv6 has been widley adopted to play that role cite IoV paper. Yet, this in itself introduced a new set of challenges, specifically for long-distance low-power communications (ie LPWAN networks) that have physical restrictions on the size of the maximum transmission unit (MTU) of which the size of an IPv6 header often exceeds. Moreover, the additional bloat introduced by the IP metadata result in longer time-on-air during message transmission thus increasing power consumption cite schc impact paper.

A solution to the aformentioned was introduced by the LPWAN working group at the IETF in the form of the SCHC protocol with mechanisms for compression and fragmentationcite rfc 8724. SCHC has recieved wide adoption, with the LoRa Allience choosing it as the IPv6 adaptation layer for LoRa based communications, as well as the emergence of a considerable range of studies examining it's use in mobility centered IoT networks, such as that of the Internet of Vehicles via IPv6 over LoRa cite IoV paper, and in Direct-to-satellite IoT networks (DtS-IoT)cite the dts-iot paper

We notice however that most of the testing done on SCHC in mobile networks required manual configuration, leaving an unexamined hole in the literature regarding the feasibility of implementing SCHC in mobile applications in which the most impactful metadata fields become uncompressable via SCHC's framework (ie: the source and destination IPv6 addresses).

This study aims to evaluate the use of weight assignment hueristics on network traffic to construct new SCHC rules based on the state of the end-to-end communication, thus providing a mechanism for dynamic rule updating.

This work is organized as follows. Chapter 2 is an overview of the SCHC framework. Chapter 3 describes the short comings of SCHC in mobile networks. Chapter

4 describes the packet scoring algorithm and the test bed used. Chapter 5 presents and discusses the results.

Chapter 2

Network Architecture - INCOMPLETE

Because SCHC is built with LPWAN in mind cite rfc 8873, there is an implicit assumption that the communication is between a low-power low-compute IoT device (DEV) and a more powerfull network gateway (Core).

Chapter 3

SCHC Compression Background

The SCHC protocol uses a lossless, context based compression mechanism, that is, it makes use of previous knowledge about a given communication to omit repeated parts of a packet's metadata. Though it differs from other context based compression protocols in the derivation of said contexts, in which contexts are acquired in a purely offline and manual manner, relying on the network admin's assumed pre-existing knowledge on the state of the communication between two endpoints to predefine the rules needed. This assumption is not only safe, given the stable and simple nature of IoT applications with devices often running a single application, but also plays an important role in bypassing the need for synchronization messages. To put it into perspective, the minimum size of the metadata in a COAP/UDP/IPv6 network stack is ≈ 60 *bytes* per packet, while a single SCHC context represented in YANG format and compressed via CBOR (the defacto-standard binary representation in IoT networks cite CBOR) results in a message of ≈ 400 *bytes*, this in itself makes flow based context synchronization (similar to that used in the ROHC protocol) unideal as we will need to compress 7 IPv6 packets per flow before seeing a net gain, which is unrealistic given that minimizing the time-on-air of communications is the main reason we perform compression in the first place.

3.1 Context Structure

Before compression, the sender must decide on a context that fits the current packet. To best understand how a context rule is choosen, it is benificial to cover the hierarchical structure of SCHC contexts.

3.1.1 Context

Given the LPWAN network structure that SCHC operates in, we have it that every DEV only stores a single context, while the Core stores contexts for all the devices connected to it. A context here is a pair $C = (CID,\ R)$ in which the CID is a unique identifier for the devices connected to the Core, used to decide on which context to use based on the DEV at the other end of the communication, the SCHC standard puts no restrictions on what can be used as a CID and provides the option to use the DEV's datalink address (ie: MAC address). R on the other hand is a set of compression rules that can be used for the communication with the DEV that the CID corresponds to.

3.1.2 Rules

Similar to Contexts, a rule is represnted as a pair $r = (RID,\ F)$ in which RID is a unique rule ID with it's uniqueness being in reference to the Core's network and by extension all the DEVs connected to the Core. While F is a set of field descriptors that dictate how every field is to be compressed.

3.1.3 Field Descriptors

Finaly, A field descriptor is a 7-tuple $f = (FID,\ FL,\ FP,\ DI,\ TV,\ MO,\ ACT)$ in which the FID is an identifier for the field's type (ie: IPv6 source address, UDP checksum, etc), FL refers to the size of the field in bits, FP is a position identifier used to distinguish between fields that repeat in the same header (ie: CoAP Uri-Path), DI is the direction of the packet as referenced in Section 2, TV is a set of target value(s) that the field is expected to have (can either be a single value or a vector), MO is a comparision operation performed on the field and the TV to decide if compression is allowed, and ACT is the compression action to perform on the field.

3.2 Packet Classification

Packet Classification refers to the two stage process of determining the subset of rules that a packet can be compressed by. The compressor iterates over all the rules in the choosen context, and for each rule it first performs a set of general checks that do not require looking into the content of the packet's fields, once the rule passes all the general checks we can then perform the matching operation which requires a deeper level of inspection on the packet's header.

3.2.1 General Checks

The compressor begins by ensuring that for every field in the packet's header there exists a corresponding field descriptor with a matching FID. It then checks that the matched field descriptors have a DI value that matches the traffic direction, that is, it is either set to "UP" for uplink traffic, "DW" for downlink traffic or "BI" for all traffic directions. Finally, the compressor checks for the existance of repeated fields in the packet's header, this check involves looking into the FP value of the matched field descriptors, if FP is set to 0 then the existance of repeated fields is not considered, thus all repeated fields are matched to the same field descriptor. On the otherhand if there exists a field descriptor $f \mid f[FP] = j > 0$ then there must exist a j^{th} repeated instace of the field in the packet's header.

The SCHC standard requires that a rule passes all the general checks, thus at any point when a check fails that rule is imediatly disregarded and the compressor moves to checking the next rule in the context.

3.2.2 Matching Operations

Once a rule passes all the general checks, we will have it that every field in the packet's header will be matched to an appropriate field descriptor in that rule. Now for every field and its matched field descriptor, the compressor will apply the matching function defined by the descriptors MO value on both the field and the descriptors target value(s). SCHC defines 4 of matching functions that the field descriptor can use:

- **Equal:** A direct equality comparision between the field's value and the descirptor's TV; requires the TV to be a single value.
- **Ignore:** No comparission is performed, always returns True.
- **MSB(x):** The compressor compares the x most significant bits in the field against the TV; requires the TV to be a single value.
- **match-mapping:** Checks if the field is **Equal** to any of the values stored at the TV; requires te TV to be a vector.

Note that the SCHC standard makes no further restrictions on rule selection beyond the afformentioned checks; observe that applying both the general checks and the matching operations does not guarentee that only a single rule is valid. Thus different implementations might want to test all the rules in the context first then choose one with the highest compression ratio, while other implementations might prefer to short circuit the checking process and choose the first matching rule.

3.3 Compression

Given a compression rule and a packet header, the compressor begins iterating over the rule's field descriptors in the order that they are declared in the rule's definition. In which it applies the compression action, ACT, of each field descriptor on it's matched field.

3.3.1 Compression Actions

Similar to matching operations, compression actions are a set of functions defined by the SCHC standard that perform the actual compression of the field values. Compression functions can either retun nothing; thus a complete compression where the field value is omitted, or they can return a residue; thus replacing the field's by a smaller value that will be used to derive the original field value when decompressing. There are 7 compression functions that a compressor can use:

- **not-sent:** The field is completly ommitted, and no compression residue produced.
- **value-sent:** The entire field is returned as is and is considered the compression residue.
- **mapping-sent:**

Chapter 4

Conclusion

Bibliography

[1] Leslie Lamport. *A Document Preparation System Latex User's Guide and Reference Manual*. Addison-Wesley, 1986.