(Demo) Joint Automated Header and Payload Compression in Constrained Networks

Ichrak Kallala, Thomas Watteyne, Quentin Lampin, Marion Dumay, Stephane Coutant, Cédric Adjih, Paul Muhlethaler

(Demo) Joint Automated Header and Payload Compression in Constrained Networks

Ichrak Kallala*, Thomas Watteyne*, Quentin Lampin†, Marion Dumay†,
Stephane Coutant†, Cedric Adjih*, Paul Muhlethaler*
* Inria, France
† Orange Innovation, Meylan, France
e-mail: first.last@{inria.fr*,orange.com†}

***Abstract*—Reducing the number of bytes transmitted by a low-power wireless device greatly reduces its power consumption. While header compression is a well-studied topic with solutions such as SCHC that are well-established standards, very little work exists on compressing the payload. This is all the stranger that the payload typically contains more bytes than the headers. This demonstration introduces Dixy, a payload compression technique which can be used alongside SCHC. We implement SCHC and Dixy on the nRF52840, a popular micro-controller. We have them compress packets collected from a real-world deployment by startup company Falco. We show how the resulting joint header and payload compression reduces the number of bytes exchanged between two boards by 74%. The demonstration allows visitors to understand SCHC and Dixy, trigger packets being compressed and transmitted, and observe the number of bytes and the charge consumed with enabling header and/or payload compression.**

***Index Terms*—constrained networking, compression, SCHC, Dixy.**

I. Introduction

The wireless radio transceiver is typically the component in a low-power wireless device which accounts for the largest charge consumed. Let's take a wireless temperature and humidity sensor composed of a state-of-the-art nRF52840 system-on-chip and an SHT31 sensor. Acquiring temperature and humidity from the sensor draws 600 μA for 2.5 mA, or **1.5** μ**C**. Preparing the data for transmission takes the 64 MHz micro-controller at most 2 ms, with a current draw of 3.3 mA, or **6.6** μ**C**. Transmitting a single 128-byte frame over its IEEE802.15.4 radio takes 4.256 ms at 6.53 mA, or **28** μ**C**. Optimizing the power consumption involves reducing the consumption of the radio.

Reducing the number of bytes means reducing the time spent transmitting, which correlates almost linearly with power consumption. A wireless frame typically consists of headers followed by a payload. The headers contain the metadata allowing the network to deliver the payload to the intended destination, and includes in particular different addressing fields. The payload is created by the source, carried as an opaque string of bytes by the network, and delivered to the destination.

Reducing the overall number of bytes can be achieved by "compressing" the headers and payload. Header compression has received a lot of attention by the scientific and standardization community, with Static Context Header Compression (SCHC) [1] being a well-established standardized compression routine. Payload compression has strangely received far less attention, even though the payload typically takes up more bytes in a frame that the headers. We have been developing an approach called Dixy based on previous work by Massey *et al.* [2].

The goal of this demonstration is to show SCHC and Dixy working together. We implement both in Python and C, and run both on a pair of communicating nRF52840-DK boards. We have one of the boards send 100 frames to the other. The packets contains both headers and payload, and are taken from a real-world deployment by startup company Falco[1]. In the demonstration, we can enable and disable SCHC and Dixy compression independently. We show how the overall number of bytes exchanged between the boards is reduced by 74% when enabling both SCHC and Dixy.

The remainder of the extended abstract is organized as follows. Section II provides the scientific background of the demonstration. Section III describes the joint SCHC and Dixy compression approach used in the demonstration. Section IV details the Falco dataset used. Section V describes the demonstration. Finally, Section VI concludes this extended abstract and presents avenues for future work.

II. Scientific Background

Compression in low-power wireless communication has been the focus of standardization for some years.

Most methods only compress headers at the medium-access, networking, and transport layers [3]. 6LoWPAN (RFC6282) [4] and 6LoWPAN-GHC (RFC7400) [5] are standards which compress IPv6 and UDP by removing bytes which have known values, or which can be recomputed based on the value of other fields. SCHC (RFC8724) [1] can be seen as a generalization of the approach. The compressor operates with a set of rules indicating how it can compress the fields from arbitrary headers. By crafting those rules, one can achieve the same compression as 6LoWPAN, but also compress a much larger set of protocol headers.

[1] www.wefalco.com

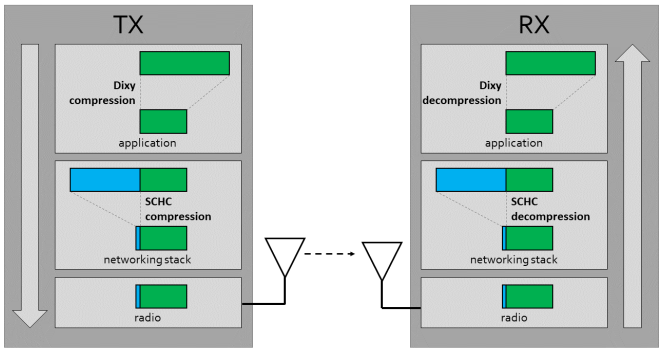

Fig. 1. We demonstrate how both the headers and the payload of a low-power wireless frame can be compressed, resulting in a reduction in bytes exchanged by 74% on the real-world dataset used.

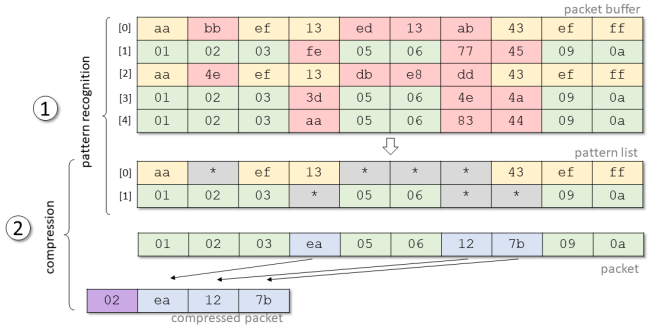

Fig. 2. Illustrating how Dixy works by presenting example contents for its internal "packet buffer" and "pattern list" structure, and how those are used to compression a 10 B packet into a 4 B compressed packet.

None of the techniques above addresses compressing the payload, despite the fact that the payload can take up more bytes in a frame than all headers. Surprisingly few studies focus on payload compression. Massey *et al.* develop a dictionary-based technique [2]: the compressor remembers the five last frames, and creates a set of patterns by identifying the bytes which are the same between these different cached frames. When a new packet is received, if the bytes from the patterns are again present, it removes those and, in a single-byte header prepended to the compressed payload, it sets as bit flag indicating this pattern was used. The power of dictionary-based compressor is that it can compress arbitrary sequences of bytes, which makes it perfect for payload compression. This technique is effective because, in a typical low-power wireless (sensor) network, each node tends to send about the same frame each time, with a few bytes difference.

We modify the technique by Massey *et al.* to create Dixy, the payload compressor used in this demo and detailed in Section III.

III. Joint Header and Payload Compression

This is arguably the first demonstration of joint compression of both the headers and the payload[2].

[2] As an online addition to this paper, the source code used in this demo will be available under an open-source MIT license at https://github.com/veryverychic/.

TABLE I
The Falco dataset.

number of devices	614 devices
duration of the recording	3.909 days
number of packets collected	1,154,176 packets
number of packets per device (min / median / max)	1,050 / 1,118 / 11,684 packets
payload length (min / median / max)	4 / 29 / 58 B

Fig. 1 shows a block diagram of the approach. The node on the left transmits (TX) frames to the node on right who receives them (RX) wirelessly. These frames are comprised of a header (blue) and a payload (green). Note that what we call "header" can be a series of headers (e.g. IEEE802.15.4 followed by IPv6 and UDP compressed using 6LoWPAN). The application on the TX side generates a payload that is compressed using Dixy. The networking stack then prepends a header to it, which itself is compressed using SCHC. The frame actually transmitted by the radio consists of the compressed header, followed by the compressed payload.

We use SCHC unmodified, as defined in RFC8724 [1]. Prior to the demonstration, SCHC rules are loaded which compress the IPv6 and UDP headers.

For compressing the payload, we modify the dictionary-based compression by Massey *et al.* [2]. The original compressor only considered patterns with consecutive bytes. In the case where successive frames are composed of mostly identical bytes, with varying bytes at different non-consecutive locations in the frame, Massey *et al.*'s approach is very inefficient. We illustrate how Dixy works in Fig. 2. It relies on a pattern buffer holding the last 5 packets. Based on that, it recognizes the patterns by identifying the sets of frames in the pattern buffer which looks the most alike. In Fig. 2, we notate bytes that are varying by a $*$. When a new packet is ready to be compressed, Dixy identifies the pattern which matches to it (here pattern 1), keeps only the bytes which are varying, and prepends a 1-byte header indicating pattern 1 was applied.

IV. Falco Dataset

Falco[3] is a start-up company which provides marinas with low-power wireless sensors. Different sensors generate different types of data and large deployments with different types of sensors producing diverse types of data.

We have one of the deployments log all (cleartext) *payloads* generated by the wireless sensors. Table I contains the high-level statistics of the Falco dataset. Over the 3.909 days the logging was active, each of the 614 devices generated between 1,050 and 11,684 payloads, with a median of 1,118, for a total of over a million payloads.

V. Demonstration

We connect two nRF52840-DK boards to a computer on which we run two interfaces, each driving one board.

[3] https://wefalco.com/en_us/

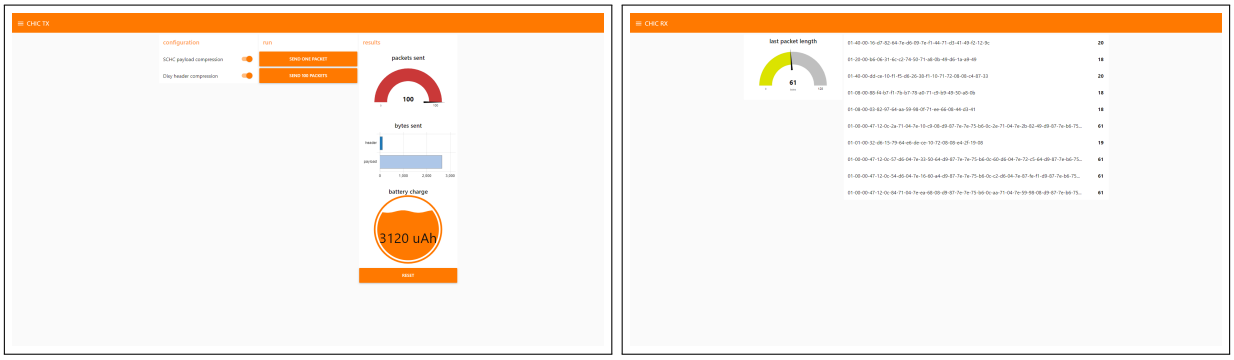

Fig. 3. Graphical user interface of the program controlling the transmitting (TX) board *(left)* and the receiving (RX) board *(right)*.

TABLE II
MEASURING THE NUMBER OF BYTES TRANSMITTED AND THE CHARGE CONSUMED AFTER SENDING 100 PACKETS FROM THE FALCO DATASET.

SCHC compression	Dixy compression	total bytes headers	total bytes payloads	total bytes		charge consumed transmitting
no	no	4,800 B	5,800 B	10,600 B		3,393 μC
yes	no	100 B	5,800 B	5,900 B	(-44%)	1,889 μC
no	yes	4,800 B	2,875 B	7,675 B	(-28%)	2,457 μC
yes	yes	100 B	2,647 B	2,747 B	(-74%)	880 μC

Fig. 3 shows screenshots of those interfaces. On the TX side, we configure whether we want to enable SCHC header compression, and/or Dixy payload compression. Two buttons allow us to send one or 100 packets. Graphical-elements allow the user to follow the activity of the TX board: the number of packets transmitted, the total number of header and payload bytes transmitted, and the charge remaining in an emulated battery initially holding 4000 μC. A button allows the user to reset these graphical elements at any time. On the RX side, the user sees the length of the last received frame. The contents of the last 10 frames received is displayed in a table.

We pick an arbitrary mote from the Falco dataset, and send the packets it generated through the demonstration system.

The emulated battery model is implemented as follows. We assume battery holding an initial charge of 4000 μC. Each time a frame is sent, we invoke function `_packet_charge()` with the total number of bytes in that frame (the sum of the header and payload bytes). Assuming a datarate of 250 kbps, we compute the time on air of that frame. Assuming a transmit current of 10 mA, we compute the charge, in Coulomb, consumed by sending that frame. This value is subtracted from the charge of the emulated battery.

Table II shows the performance of joint header and payload compression on the demonstration system. We can see that enabling only SCHC (resp. only Dixy), reduces the total bytes sent by 44% (resp. 28%). Enabling both SCHC and Dixy reduces the number of bytes by 74%. This results in a reduction of the charge consumed from 3,393 μC to 880 μC, compared to when no compression is used.

VI. CONCLUSION

This is arguably the first demonstration of joint header and payload compression in low-power wireless networks. While header compression is a well-studied topic with solutions such as SCHC being well-recognized standards, strangely, very little research has been done on compressing the payload. This is all the stranger that a frame can contain more payload bytes than header bytes. We develop Dixy, a payload dictionary-based compression technique by generalizing previous work by Massey *et al.*. To measure the performance of the resulting joint header and payload compression, we collect packets from a real-world deployment by startup Falco. We create a demonstration system where we inject packets from that dataset in a transmitter board, which compresses the payload and the header, and sends that wirelessly to a receiver board. We show how our joint header and payload compression results in 74% less bytes being exchanged.

REFERENCES

[1] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J. Zuniga, *SCHC: Generic Framework for Static Context Header Compression and Fragmentation*, Internet Engineering Task Force (IETF) Std. RFC8724, 2020.

[2] T. Massey, A. Mehta, T. Watteyne, and K. Pister, "Protocol-Agnostic Compression for Resource-Constrained Wireless Networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, Miami, Florida, USA, 6-10 December 2010.

[3] M. Tömösközi, M. Reisslein, and F. H. P. Fitzek, "Packet Header Compression: A Principle-Based Survey of Standards and Recent Research Studies," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 698–740, 2022.

[4] J. Hui and P. Thubert, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*, Internet Engineering Task Force (IETF) Std. RFC6282, 2011.

[5] C. Bormann, *6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, Internet Engineering Task Force (IETF) Std. RFC7400, 2014.