Sensors & Transducers, Vol. 196, Issue 1, January 2016, pp. 43-51

Sensors & Transducers

http://www.sensorsportal.com

Packet Header Compression for the Internet of Things

*** Pekka KOSKELA, Mikko MAJANEN and Mikko VALTA**
VTT Technical Research Centre of Finland Ltd, P.O. Box 1100, FI-90571 Oulu, Finland
* Tel.: +35 8 40 751 390, fax: +35 8 20 722 2320
* E-mail: pekka.koskela@vtt.fi

Received: 15 December 2015 /Accepted: 18 January 2016 /Published: 31 January 2016

Abstract: Due to the extensive growth of Internet of Things (IoT), the number of wireless devices connected to the Internet is forecasted to grow to 26 billion units installed in 2020. This will challenge both the energy efficiency of wireless battery powered devices and the bandwidth of wireless networks. One solution for both challenges could be to utilize packet header compression. This paper reviews different packet compression, and especially packet header compression, methods and studies the performance of Robust Header Compression (ROHC) in low speed radio networks such as XBEE, and in high speed radio networks such as LTE and WLAN. In all networks, the compressing and decompressing processing causes extra delay and power consumption, but in low speed networks, energy can still be saved due to the shorter transmission time. *Copyright © 2016 IFSA Publishing, S. L.*

Keywords: Internet of Things (IoT), Compression, Energy efficiency, Wireless radios.

1. Introduction

Due to the extensive growth of Internet of Things (IoT), the number of wireless devices connected to the Internet is increasing and will continue to increase remarkably in the near future. For example, Gartner estimates that the IoT, which excludes PCs, tablets and smartphones, will grow to 26 billion units installed in 2020, representing an almost 30-fold increase from 0.9 billion in 2009 [1].

In wireless IoT networks, the available bandwidth and energy is often restricted. Therefore, all means for saving those resources are welcome. The biggest resource consumption source is radio communication, including both transmission and reception [2]. One efficient way to control transmission and reception is the utilization of the duty-sleep cycle control of the radio with a MAC protocol [2]. Another effective way to reduce radio transmission, which supplements the previous approach, is the utilization of packet compression. The packet compression can be targeted to cover only header or payload part, or both parts of the packet.

IoT devices, e.g., sensors, periodically report their current data values to the cloud services in the Internet. Thus, the transmitted data may be only couple of bytes, whereas the protocol headers of the packet (MAC, IP, TCP/UDP, etc.), are many tens of bytes. This big header overhead is the motivation behind the compression of packet headers and the design of lightweight protocols with small headers. For instance, the header of the traditional application layer protocol, Hypertext Transfer Protocol (HTTP), can take easily over 40 bytes, whereas replacing HTTP with Constrained Application Protocol (CoAP) [3] can drop the header size to less than 10 bytes.

In this paper, we first give a brief description of packet compression techniques in Section 2, which is followed by a more careful presentation of the evolution of header compression. In Section 4, we present the results of ROHC header compression for CoAP/UDP/IP protocols in the case of both fast

(LTE, WLAN) and low speed (XBEE) radio networks. Finally, Section 5 concludes the paper.

2. Packet Compression Techniques

There are several techniques to compress packets, see for example recent surveys in [4-7]. In this section, we shortly summarize the main compression techniques that are depicted in Fig. 1.

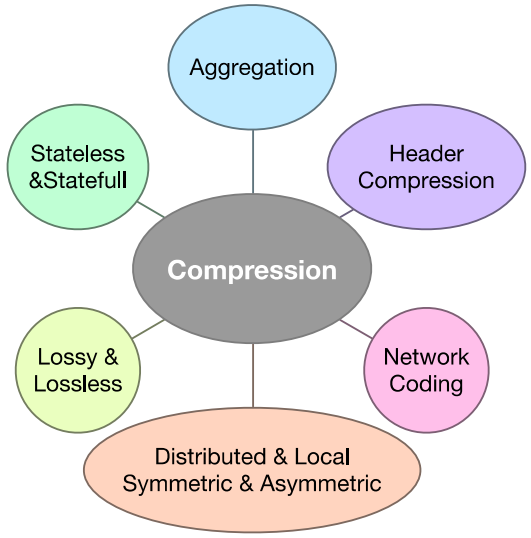

Fig. 1. Compression techniques.

2.1. Aggregation

Aggregation is an efficient method to compress data whenever it is suitable for the function in question. The problem of using aggregation is that it will always reduce information like by averaging, and the network structure and routing has to support aggregation. The other shortage is that if packet loss exists, then losing one aggregated packet means losing several original packets, especially when no packet caching is used. During decades, numerous aggregation schemes have been proposed, in which the most recent ones, e.g. [8] and [9], consider also security issues.

2.2. Network Coding

Depending on the data information, there may be a possibility to define sequences, which are repeating inside the message(s). By utilizing the repeated information, it is possible to describe the same information in a shorter way and to achieve compression in that way.

In traditional routing networks, packets are cached and forwarded separately downstream, even if they have the same destination. In network coding, the messages are merged and the code and the accumulated result are forwarded to the destination. After receiving the accumulated message, it is decoded at the destination. Network coding techniques for wireless sensor networks (WSN) are discussed more carefully in [10].

There are several methods with varied complexity for describing the repeated information. Depending on the method, the decompression will return exactly the original data or it may have some losses or mistakes. In general, the method with more complexity will provide more lossless compression and better compression rate.

2.3. Distributed vs. Local and Symmetric vs. Asymmetric

Compression operation can be carried out locally at the node or it can be distributed to several nodes in order to share the load between the nodes. Load sharing may help especially nodes that have scarce of resources. Usually, capabilities of network devices vary. Terminal nodes, like sensor nodes, have least resources whereas core network devices, like servers, can share their resources. A special case of distribution is an asymmetric system, where the most capable devices take care of the highest load and thus save the resources of the terminal nodes.

2.4. Lossy & Lossless

Depending on the data reconstruction after the decompression, compression methods can be divided into lossy and lossless techniques. Lossless techniques aim to return exactly the original data, whereas lossy techniques give only an approximation of the original data. Generally, lossy algorithms provide higher compression, but also higher information loss. Which approach will fit best depends on the requirements of the application. For instance, in video and voice applications, lossy compression may be accepted, but data loss or a wrong value may cause serious problems in the case of control measurements.

2.5. Stateless or Stateful

Stateless compression does not require any per-flow state, which could be corrupted during the change in wireless connection. The idea of the stateless compression is based on the assumption that some content between the sender and the receiver is well-known and thus can be assumed or extrapolated from the received information. That kind of information is for instance the static parts of the protocols' header information, like the network prefix, which can be compressed into a single bit. If similar information is already available in several protocol headers, like checksum, then that information can be removed.

In the case of stateful compression (or shared-context), there will always be negotiation between the sender and the receiver. During the negotiation, the sender and the receiver agree on the semantics how the compression will be performed. When the compression is used, the sender and the receiver have to agree from time to time that the compression state is still valid.

The advantage of the stateful approach compared to the stateless is that it will allow much higher compression rate than the stateless. For instance, in a data flow, almost all header information can be compressed under one ID flag in the stateful approach, which is not always possible in the stateless case. Another advantage of the stateful approach is that it is more dynamic because there is no need to assume anything before the negotiation and so the packet formats can change freely.

Because of the nature of wireless communication, there will always be some packet loss and bit errors during messaging. If the losses and errors are significant, then stateless approach will outperform stateful, because the stateless approach does not need any pre-configuration before compression. In the case of stateful compression, if an error or a loss happens, as it often does in mobile ad hoc networks, then pre-configuration must be done again and again, which causes extra control traffic. So, maintaining the flow states will be difficult in a mobile ad hoc environment [11]. This makes pure stateful solution, despite it has a better compression rate, applicable only in good link conditions.

Resource-constrained devices must also consider memory usage and computational complexity. From this point of view, the stateless approach is more efficient, because it does not need any state establishment or management, and it has a simpler source code.

2.6. Non-adaptive and Adaptive

In general, an adaptive compression can adjust system to environmental changes, which can be for example changes in the data type or connection performance. The connection can be improved, for instance, by chancing the communication interface or operation of the link layer (L2), or utilizing multipath routing to get better connection. Adaptivity makes the system more complex, but at the same time, it provides better performance when the system is changing.

2.7. Header Compression

The network packet can be divided into two parts: the header information part coming from different protocols and the payload part containing the data. From the compression point of view, the header part is interesting because there is redundant information among different protocol headers and, especially, between consecutive packets belonging to the same flow. This kind of redundant information can be elided [6, 12]. For example, many header fields remain constant between packets, or change according to a known pattern. Over a single link, not all that information is needed and part of it can be temporarily removed, i.e., the full IP packet will be re-created on the receiving side of the link. In the next section, we will take a closer look to different header compression solutions.

3. Development of Header Compression

Header compression is not a new idea, but compression standards are still evolving. The first header compression scheme, CTCP (i.e., VJ compression) [13], compresses TCP/IPv4 headers and it was presented in 1990. The evolution continued when IPHC [14] and CRTP [15] were presented in 1999 with wider protocol support (UDP, RTP and IPv6) and improvements in packet loss handling. The next evolution step was presenting ROHC [16] in 2001 and 6LoWPAN [17] in 2007. ROHC presents a robust compression scheme with modular protocol support, i.e., protocol profiles. 6LoWPAN presents a compact solution applicable only for wireless IEEE 802.15.4 technology. In the next chapters, we will discuss more carefully the above mentioned header compression solutions.

CTCP *(Van Jacobson Header Compression):* CTCP header compression defines compression for TCP/IP(v4) datagrams, which was specifically designed to improve TCP/IP performance over slow serial links, with speed around 300 bps. The primary idea behind the compression was to define per-packet information on which bytes are likely to stay constant over the lifetime of a connection, and which bytes are likely to change during the connection but do not all change at the same time. In the compression, the constant bytes can be omitted and the changed bytes can be indicated with a bitmask. The bitmask tells the difference between the previous and the current packet and that way only the differences in the changing fields are sent rather than the whole fields themselves. In practise, this is done by saving the states of the TCP connections at the both ends of the link, and sending only the differences in the header fields that changed. Van Jacobson compression reduces the normal 40 byte TCP/IPv4 packet headers down to 3-4 bytes in average, see Fig. 2. Because the scheme is designed only for low bandwidth connections, where bit errors and packet loss are not an issue, it works well there. When the bit error rate (BER) increases over 10^{-4}, the scheme does not perform well [18]. This is mainly due to that the scheme does not have its own feedback and recovery mechanism concerning packet loss, instead it relies on TCP's own recovery mechanisms. Nowadays, it is well known that TCP's recovery mechanisms for packet loss do not perform well in wireless connections [19].

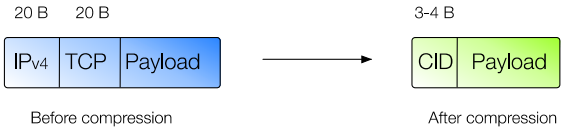

Fig. 2. CTCP header compression.

IPHC *(IP Header Compression):* In general, IPHC is based on a similar compression and decompression idea as CTCP, where known information is compressed and decompression is done based on the saved context information of the compression. The main development step that IPHC brought in was that it supports UDP, IPv6 and extension headers in addition to TCP. Moreover, IPHC improved error recovery mechanisms (important especially in lossy links) and supported multiple IP header compression (in case of tunnelling of IP packets) and allowed extensions for multi-access links and multicast. Two additional mechanisms that increase the efficiency of header compression over lossy links were also described. For non-TCP packets, compression slow-start and periodic header refreshments allow minimal periods of packet discarding after loss of a header that changes the context. When many packet streams (several hundreds) traverse the link, a phenomenon known as context ID (CID) thrashing can occur. In CID thrashing, headers cannot be matched with an existing context and have to be sent uncompressed or as full headers.

C*RTP* *(Compressing RTP Headers):* CRTP uses similar compression approach like CTCP and IPHC. As a new thing, it provides support for RTP protocol. As additional features, CRTP brings in a method for reporting packet loss information. The information contains a health report of packets and compression level for sources capable adapting. CRTP replaces the IPv4, UDP, and RTP headers (40 bytes) with a 2-4-byte context ID (CID), as depicted in Fig. 3.

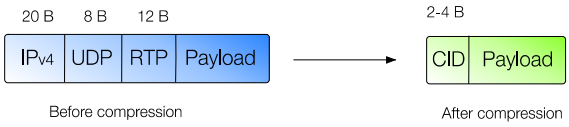

Fig. 3. CRTP header compression.

ROHC *(Robust Header Compression)* has similar idea like above, where compressed packets are decompressed based on the saved context information in the decompression side. One main difference for previous compression schemes is that the whole compression process is divided into different states depending on the link performance. The states are maintained within two finite state machines: one as a compressor and the other as a decompressor. The compressor states are Initialization & Refresh (IR), First Order (FO) (i.e., partial compression), and Second Order (SO) (i.e., full compression). The state machine aided compression makes compression process more robust and takes advantage of the link quality, but on the other hand, it increases complexity. The other remarkable difference is that new protocol headers can be presented as profiles, which makes a modular base for protocol implementation and development. Currently, ROHC supports, e.g., RTP, UDP, UDP-Lite, TCP, ESP, and IP protocols [20, 21, 22]. CoAP compression profile for ROHC was introduced in our earlier work [23]. Fig. 4 presents an example of CoAP/UDP/IPv4 packet header compression, where the headers are compressed from 37 bytes to 5 bytes.

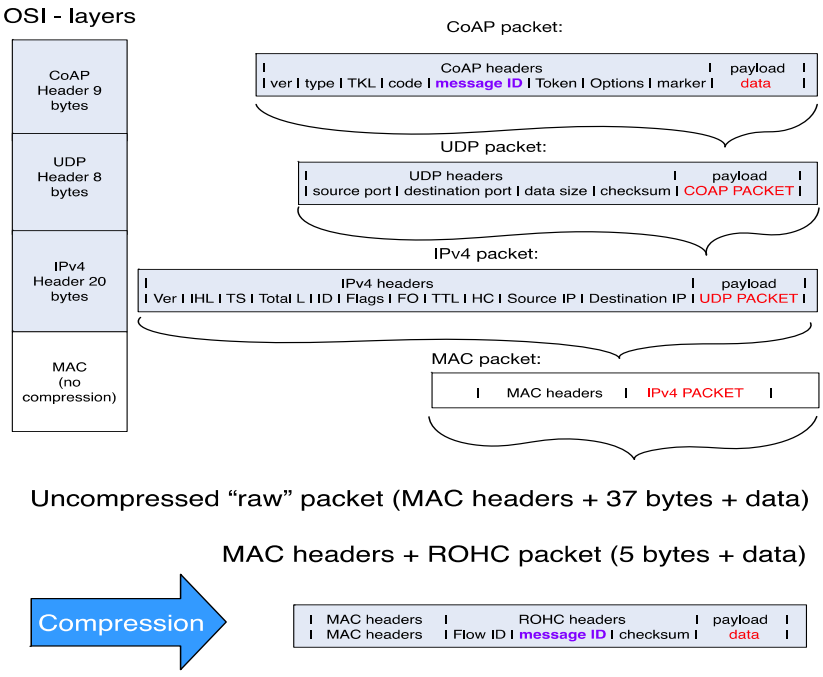

Fig. 4. ROHC header compression with CoAP/UDP/ IPv4 profile.

6LoWPAN *(IPv6 over low power wireless area networks)* compression approach differs from previous ones as being a stateless compression. It uses only known or assumed headers in the compression process and thus there is no need to create decompression tables in the receiver side. This simplifies the compression process and allows stateless compression, which has a clear advantage in lossy communication like wireless communication with resource scare devices. 6LoWPAN is not only about header compression but also involves fragmentation and reassembly. Currently, 6LoWPAN supports only IEEE 802.15.4 devices. Based on the protocol stack, 6LoWPAN header compression involves the following header compression:

- Application layer header compression
- Transport layer header compression (TCP, UDP and ICMP)
- Network layer header compression (IPv6 unicast and multicast, routing and other extension headers)
- Adaptation layer header compression (fragmentation, compression, mesh routing and IP routing headers).

6LoWPAN compresses IPv6/UDP headers (48 bytes) to 7 bytes compressed header (CH) and, respectively, IPv6/TCP headers (60 bytes) to 7-31 bytes compressed header, as depicted in Fig. 5.

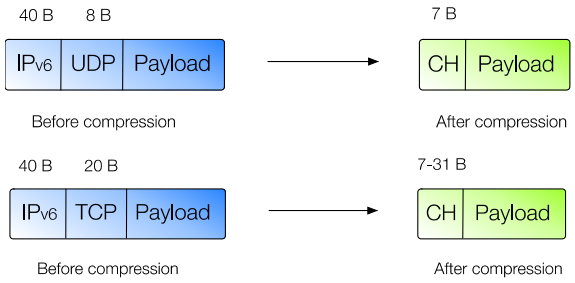

Fig. 5. 6LoWPAN header compression.

4. IoT Compression Solution with ROHC

Our IoT packet compression solution utilizes ROHC and CoAP. In our earlier study [23], we extended ROHC by presenting CoAP compression profile for ROHC. The main result of our earlier study was that the CoAP packet size could be reduced by over 90 % at best. However, compression and decompression requires processing power, and that means increase in the delay. In our tests on Raspberry Pi (RPi) computer, the compression and decompression took about 1-3 ms depending on the compression state [23]. In the case of WLAN radio, the extra processing for compression and decompression outperformed the energy savings achieved during the shorter transmission time. However, smaller packet size and shorter transmission time can reduce packet loss in lossy links with high bit error rate. That reduces the need for retransmissions, which improves the energy efficiency. In this paper, we extend our earlier work to find out the performance of ROHC compression with the CoAP profile in LTE and XBEE networks.

4.1. Test Bed and Testing Scenario

The test bed consists of a laptop and Model B+ Raspberry Pi (RPi) computers. A transmission link between RPi and laptop was made with XBEE, LTE and WLAN radios. VTT's CNL laboratory's Willab network was used to make it possible to connect RPi to the WLAN or LTE base station. In the XBEE test, the test link was made directly between RPi and laptop. The power consumption was monitored from RPi's main power supply by using a current probe and an oscilloscope. During the test, the network traffic and power consumption data was logged.

The XBee test bed is shown in Fig. 6(a). The XBee radio shield was connected to RPi's GPIO port. To make it possible to transfer IP packets through XBee network, the XBee-Tunnel-Daemon (libgbee, http://sourceforge.net/projects/libgbee/) was used. Libgbee is a daemon that creates a virtual network interface allowing transmission of UDP/IP packets over the XBee module.

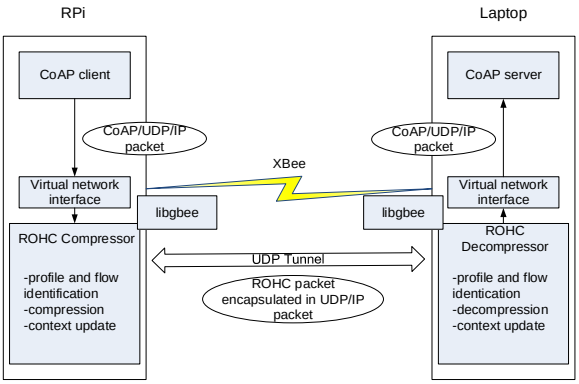

Fig. 6 (a). XBee test bed.

In the LTE radio scenario, the radio stick was connected to USB port (see Fig. 6(b)).

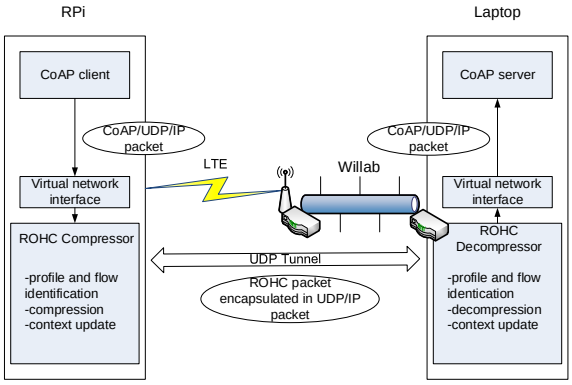

Fig. 6 (b). LTE test bed.

The LTE base station was connected to the Willab network. The WLAN test bed is described in more details in our earlier publication [23]. For the test traffic, we used the same CoAP messages as in our earlier study for WLAN in [23].

4.2. Header Compression Results

Header compression reduces the size of the transmitted packets. During the transmission, these smaller packets have lower probability to have bit errors, and hence the packet loss will be smaller than for packets with the original size [23]. Thus, packet loss and response time decrease, which together provide better performance. This will realize especially in wireless and low bandwidth links, where bit errors and packet loss are common. The header compression is beneficial when the data amount is small compared to the header part, as illustrated in Fig. 7. The figure presents ROHC compression gain with UDP/IPv4, UDP/IPv6 and "theoretical" packet as the payload increases. In the theoretical packet, it is assumed that all headers (CoAP, UDP, IP, and also MAC header) are compressed to 4 bytes.

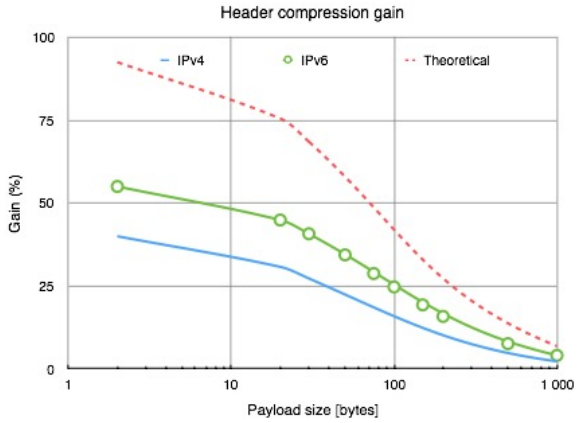

Fig. 7. Effect of payload size in CoAP packet for header compression gain with different profiles.

It can be seen that with small payloads the gain is at least 40 %. After 20 bytes of payload, the gain starts to decrease more rapidly. When the payload is over several hundreds of bytes, the advantage of the header compression gradually disappears. However, if the payload was also compressed, the header compression could become beneficial again.

Let's take a more careful look how compression affects to processing delay and radio transmission time in the case of a low bandwidth XBEE link. The whole process consists of packet creating and compression, transmission, and receiving and decompression. The compression and decompression processes create delay compared to uncompressed packet processing, as can be seen in Fig. 8.

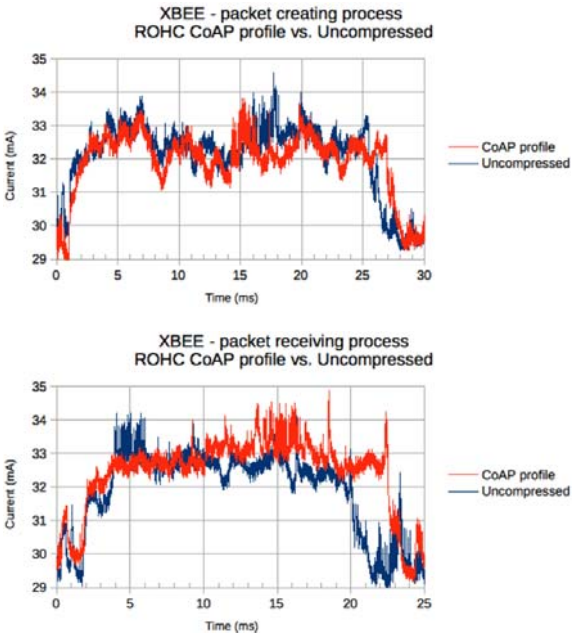

Fig. 8. CoAP/UDP/IPv4 packet creating and receiving processes in RPi using XBEE radio.

The delays are 1.5 ms and 4.0 ms in compression and decompression phases, respectively. On the other hand, because of the smaller packets, compression will speed up radio transmission by about 1.2 ms, see Fig. 9. When the delays are calculated together, the total delay remains at 4.3 ms.

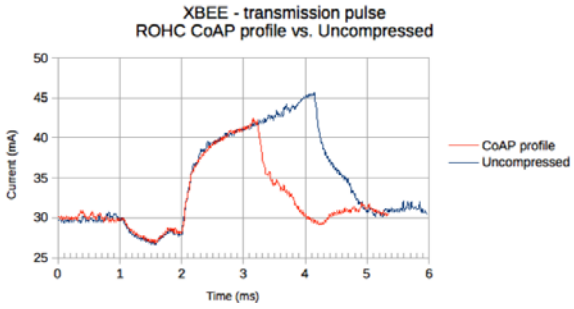

Fig. 9. XBEE transmission pulse when using ROHC with CoAP and Uncompressed profile.

Using Fig. 8 and Fig. 9, we can calculate the energy consumption by multiplying the time and the average power consumption during the packet creating & compression, transmission and receiving & decompression processes. Table 1 presents the total energy consumption in the case of CoAP/UDP/IPv4 packets utilising ROHC's CoAP profile in SO, i.e., full compression state. Even if the time saved in the transmission was shorter than the time spent for compression/decompression processing, the total energy efficiency improved by 15.1 μJ/packet (0.184 %), when the packets were compressed. This is because during the transmission, the power consumption is on a much higher level

than during packet processing. The trade-off for saving bandwidth and for lower energy consumption was the extra delay of 4.3 ms.

Table 1. Energy consumption in XBEE using ROHC's CoAP/UDP/IPv4 compression profile.

Packet processing	Uncomp. packet [ms]	Comp. packet [ms]	Time gap [ms]	Energy gap (µJ)
Creating	26.5	28	-1.5	-14.7
Receiving	22	26	-4	-40.0
Transmission	3	1.8	1.2	68.9
TOTAL			-4.3	15.1

As depicted in Table 2, transmission in LTE and WLAN is more than 200 time faster than in XBEE transmission. Thus, we could not detect the transmission peaks in our measurements, only delays of compression and decompression processing could be found out, see Fig. 10 and Fig. 11.

Table 2. Transmission times with different radios for compressed and uncompressed packet.

Radio	Transmission speed (Mbit/s)	Transmission time (µs)	
		Uncomp. packet (736 bits)	Comp. packet (440 bits)
XBEE	0.25	2944	1184
WLAN 802.11b	54	14	8
LTE	100	7	4

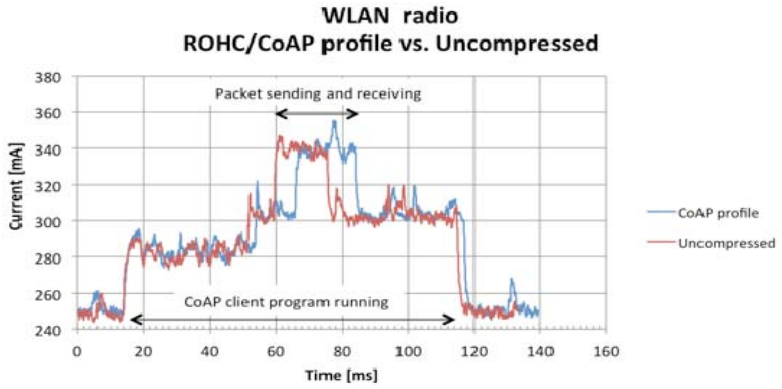

Fig. 10. ROHC's CoAP/UDP/IPv4 compression profile vs. uncompressed profile in WLAN [21].

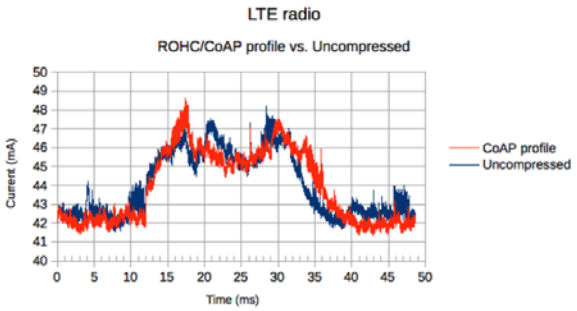

Fig. 11. ROHC's CoAP/UDP/IPv4 compression profile vs. uncompressed profile in LTE.

So, in the case of fast speed radios like LTE and WLAN, direct energy savings during transmission will be neglible and the cost of extra compression/decompression processing will rule and the total process will consume more energy than without compression. However, in lossy links, packet loss will decrease when the packet size decreases [24], i.e., when the packet is compressed. This will enhance transmission by avoiding retransmissions, and that way energy could be saved. However, the potential savings depend on how many packet losses actually can be avoided.

5. Discussion and Conclusions

Due to the extensive growth of Internet of Things (IoT), the number of wireless devices connected to the Internet is forecasted to grow to 26 billion units installed in 2020 representing an almost 30-fold increase from 0.9 billion in 2009 [1].

In wireless IoT networks, the available bandwidth and energy is often restricted. That resource can be effectively saved by reducing radio operation: transmission and reception. In the case of IoT devices like sensors, the transmitted data may be only couple of bytes, whereas the protocol headers of the packet (MAC, IP, TCP/UDP, CoAP, etc.) are many tens of bytes. This big header overhead serves as the motivation behind compressing the packet headers and that way decreasing the radio operation times.

The performance of ROHC header compression was studied with a low speed XBEE radio, and high speed LTE and WLAN radios in a real test bed environment. In our previous studies with WLAN radio, we found out that ROHC with CoAP compression profile can decrease the packet size by 90 % or more. The smaller packet size speeds up the radio operation during transmission and reception. This reduces the energy consumption during the radio

transmission. Smaller packet will also reduce packet loss in lossy links, which can further improve the energy efficiency. The trade-off for smaller packet size is the delay and extra processing power needed for the compression and decompression.

In the case of XBEE, the energy saving during radio transmission was bigger than energy consumption during compression/decompression processes. The total energy savings were 15.1 μJ/packet (0.184 %). The compression and decompression processes themselves consume roughly ten times less energy than the whole packet creating and receiving processes. However, the delay caused by the compression and decompression was bigger than the saved time during the radio transmission, so the trade-off for smaller packet and energy savings was the extra delay of 4.3 ms.

In the case of high speed radios (LTE and WLAN), the savings in transmission time and energy were neglible compared to the extra processing needed for compression and decompression. Thus, there were no energy savings during individual packet transmissions. However, the smaller packet size will reduce packet loss in lossy links and this way it could be possible to save energy, enhance throughput and decrease delay.

The compression and decompression processes cause quite long delays. They can possible be decreased by having a more optimal software implementation. Reducing the delay caused by the compression/decompression processing automatically improves also the energy efficiency.

ROHC has good compression gain and it is used in current mobile networks, which make it a promising candidate for IoT header compression solution. The disadvantage is the stateful operation and more complexity compared to 6LowPAN. One possible future work item could be to merge 6LowPAN functionalities to ROHC to have some stateless compression at the Initialization and Refresh (IR) state, and perhaps to extend the compression profiles to cover also MAC headers like 802.15.4 and Bluetooth Low Energy.

Acknowledgements

This work was supported by TEKES as part of the Internet of Things program of DIGILE.

References

[1]. Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020, (http://www.gartner.com/newsroom/id/2636073).

[2]. P. Koskela, M. Valta, T. Frantti, Energy Efficient MAC for Wireless Sensor Networks, *Sensors & Transducers Journal*, Vol. 121, Issue 10, October 2010, pp. 133-143.

[3]. Z. Shelby, K. Hartke, C. Bormann, The Constrained Application Protocol (CoAP), *Internet Engineering Task Force (IETF)*, Request for Comments: 7252, Category: Standards Track, 2014, pp. 1-112.

[4]. P. Koskela, M. Majanen, Robust Header Compression for Constrained Application Protocol, *Internet of Things Magazine Finland*, No. 1, 2014, pp. 36-39. http://www.internetofthings.fi/extras/IoTMagazine2014.pdf

[5]. Razzaque M. A., Bleakley C., Dobson S., Compression in wireless sensor networks: A survey and comparative evaluation, *ACM Transaction on Sensor Networks*, Vol. 10, No. 1, 2013, pp. 5-1–5-44.

[6] Shivare M. R., Maravi Y. P. S., Sharma S., Analysis of Header Compression Techniques for Networks: A Review, *International Journal of Computer Applications*, Vol. 80, No. 5, 2013, pp. 13-20.

[7]. T. Srisooksai, K. Keamarungsi, P. Lamsrichan, Araki K., Practical data compression in wireless sensor networks, *Journal of Network and Computer Applications*, Vol. 35, No. 1, 2012, pp. 37-59.

[8]. L. Yu, J. Li, S. Cheng, S. Xiong, H. Shen, Secure Continuous Aggregation in Wireless Sensor Networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 3, 2014, pp. 762-774.

[9]. S. Roy, M. Conti, S. Setia, S. Jajodia, Secure Data Aggregation in Wireless Sensor Networks: Filtering out the Attacker's Impact, *IEEE Transactions on Information Forensics and Security*, Vol. 9, No. 4, 2014, pp. 681-694.

[10]. P. Ostovari, J. Wu, A. Khreishah, Network coding techniques for wireless and sensor networks, in The Art of Wireless Sensor Networks, H. M. Ammari (Ed.), *Springer*, 2013, pp. 1-35.

[11]. B.-N. Cheng, J. Zuena, J. Wheeler, S. Moore, B. Hung, MANET IP Header Compression, in *Proceedings of the IEEE Military Communications Conference (MILCOM'13)*, 2013, pp. 494-503.

[12]. Effnet AB WHITE PAPER Library, An introduction to IP header compression, 2004. (http://www.effnet.com/sites/effnet/pdf/uk/Whitepaper_Header_Compression.pdf).

[13]. V. Jacobson, Compressing TCP/IP Headers, *Internet Engineering Task Force (IETF)*, Request for Comments: 1144, Category: Standards Track, 1990, pp. 1-46.

[14]. M. Degermark, B. Nordgren, S. Pink, IP Header Compression, *Internet Engineering Task Force (IETF)*, Request for Comments: 2507, Category: Standards Track, 1999, pp. 1-47.

[15]. S. Casner, V. Jacobson, Compressing IP/UDP/RTP Headers for Low-Speed Serial Links, *Internet Engineering Task Force (IETF)*, Request for Comments: 2508, Category: Standards Track, 1999, pp. 1-24.

[16]. C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, H. Zheng, RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP and uncompressed, *Internet Engineering Task Force (IETF)*, Request for Comments: 3095, Category: Standards Track, 2001, pp. 1-168.

[17]. G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks, *Internet Engineering Task Force (IETF)*,

Request for Comments: 4944, Category: Standards Track, 2007, pp. 1-30.
[18]. M. Degermark, H. Hannu, L. Jonsson, K. Svanbro, Evaluation of CRTP performance over cellular radio links, in *IEEE Personal Communications Magazine*, Vol. 7, No. 4, Aug 2000, pp. 20-25.
[19]. Tian Ye, K. Xu, N. Ansari, TCP in wireless environments: problems and solutions, in *IEEE Communications Magazine*, Vol. 43, No. 3, March 2005, pp. 27-32.
[20]. Pelletier G., Sandlund K., RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite, *Internet Engineering Task Force (IETF)*, Network Working Group, Category: Standards Track, Request for Comments: 5225, 2008, pp. 1-124.
[21]. K. Sandlund, G. Pelletier, L.-E. Jonsson, The RObust Header Compression (ROHC) Framework, *Internet Engineering Task Force (IETF)*, Category: Standards Track, Request for Comments: 5795, 2010, pp. 1-41.
[22]. G. Pelletier, K. Sandlund, L.-E. Jonsson, M. West, RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP), *Internet Engineering Task Force (IETF)*, Category: Standards Track, Request for Comments: 6846, 2013, pp. 1-96.
[23]. M. Majanen, P. Koskela, M. Valta, Constrained Application Protocol Profile for Robust Header Compression Framework, in *Proceedings of the Fifth International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (ENERGY'15)*, Rome, Italy, 24-29 May 2015, pp. 47-53.
[24]. N. Golmie, Coexistence in Wireless Networks: Challenges and System-Level Solutions in the unlicensed band, *Cambridge University Press*, Cambridge, 2006, pp. 40-41.