Implementation and Evaluation of a TCP Header Compression for 6LoWPAN

Ahmed Ayadi, Patrick Maillé, David Ros, Laurent Toutain and Tiancong Zheng
Institut Télécom / Télécom Bretagne
Rue de la Châtaigneraie, CS 17607
35576 Cesson Sévigné cedex, France
Email: {firstname.lastname}@telecom-bretagne.eu

***Abstract*—This paper presents an experimental study of TCPHC, a novel algorithm for compressing the Transmission Control Protocol (TCP) header to reduce its overhead in IPv6-enabled Low-power and Lossy Networks (6LoWPANs). Results show that TCPHC outperforms TCP both in low-loss and high-loss networks. In fact, TCPHC can reduce the TCP header to 6 bytes in more than 95% of the cases. Moreover, experimental results show that our TCP header compression algorithm reduces the energy consumption by up to 15%.**

***Index Terms*—Low-power and Lossy Networks, 6LoWPAN, TCP, Energy-Efficiency, Header Compression**

I. Introduction

Nowadays, Low power and Lossy Networks (LLNs) are becoming more and more popular and they are currently used in many industrial fields such as tracking systems [1], health monitoring [2], etc.

Currently, most LLN devices use low-power wireless cards such as Bluetooth, Low power IEEE 802.11, and especially IEEE 802.15.4. Since the introduction of IPv6 to these devices, they are able to communicate not only between each other but also potentially with every IP device. Those new devices, called IP Smart Objects[1], are changing the concept of the Internet, to be not limited to a system of interconnected computer networks, but expand to an interconnection of all daily life objects leading to the concept of the *Internet of things* [3].

In that promising context, three IETF working groups concerning LLN networks have recently been set up, such as IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), Routing Over Low power and Lossy networks (ROLL[2]) and Constrained Restful Environments (CoRE[3]). The 6LoWPAN Working Group [4] (WG) focuses on neighbor discovery and header compression. It has proposed LOWPAN_IPHC (IPHC) [5], a new version of the LOWPAN_HC1 IPv6 header compression mechanism [6], which can reduce the IPv6 header to about 3-5 bytes. That group has specified in the same document LOWPAN_NHC, a header compression mechanism for the transport layer. In [5], only a UDP [7] datagram compression mechanism is specified. UDP header compression is useful for 6LoWPANs because many applications are fault-tolerant and do not require full reliability from the transport protocol. However, other applications and services (such as SSH and HTTP) are not fault-tolerant. Those kinds of applications require a reliable service which UDP cannot provide. Moreover, some LLNs' application areas, such as health, military and security applications, impose strong reliability constraints. In some usage cases (e.g., sending a software update to a wireless node, or sending a query requesting specific information from the wireless node), there is a need for a reliable data transport.

In this paper, we focus on TCP over LLNs, since TCP is the most used reliable transport protocol over Internet. The TCP usage allows current IP-based devices to communicate directly with LoWPAN devices using their TCP/IP stack. Moreover, non-TCP/IP reliable transport protocols need a complex algorithm on the Edge Router. Thus the use of TCP reduces the complexity at the Edge Router (ER), which is an IPv6 router that interconnects the 6LoWPAN to another IP network.

Nevertheless, TCP performance is mainly harmed due to its header length. As an illustration, the maximum physical layer packet size in IEEE 802.15.4 networks [8] is 127 bytes, and medium access layer and link layer security requirements leave only 81 bytes for upper layers data; finally, the use of IPv6 [9] as a network protocol header (40 bytes) without compression leaves only 41 bytes for transport protocols. TCP uses 20 bytes in the header (if no TCP options are included), which leaves only 21 bytes for the application layer if no compression scheme is applied.

Likewise, a pure TCP acknowledgment (i.e., a TCP ACK carrying no data) without any TCP options represents 25% of the payload of an IEEE 802.15.4 MAC frame while TCP pure ACKs represent roughly 33% of the total number of segments exchanged in a TCP session (this figure may go up to roughly 50% if the Delayed ACK mechanism is not used).

Therefore, some TCP header compression 6LoWPANs. Such a compression algorithm should respect some requirements: efficiency (the scheme must provide low overhead in all cases), transparency (the resulting header after a compression and decompression should be identical to the original header), and disordering tolerance (the scheme must be able to decompress compressed segments correctly even when segments

[1] http://ipso-alliance.org/
[2] http://datatracker.ietf.org/wg/roll/charter
[3] http://datatracker.ietf.org/wg/core/charter

978-1-4577-9538-2/11/$26.00 ©2011 IEEE

arrive with a moderate disordering (1-2 packets)) .

We consider here LOWPAN_IPHC (TCPHC) [10], a new TCP header compression algorithm recently introduced to reduce significantly the TCP header size for 6LoWPAN. The TCP header compression is performed in the edge router between the 6LoWPAN and the external IP network. Moreover, the TCPHC mechanism can be used in conjunction with IPHC [5] and thus reduces all overheads to about seven to 10 bytes instead of 60 bytes.

The goal of this paper is to give an experimental evaluation of the TCPHC algorithm. The evaluation is done on our testbed in different environments (low-lossy and high lossy environments). We compare the performance of the legacy TCP and of TCPHC based on two main metrics, namely the transfer duration and consumed energy. The results show that the TCPHC mechanism can reduce the size of the TCP header to 6 octets in 95% of the cases and the consumed energy by 9% to 16% depending on the TCP scenario.

The remainder of this paper is organized as follows. Section II presents a brief overview of the related works in the area of TCP header compression in LLNs. Section III presents an overview of the TCPHC mechanism. We describe our testbed in Section IV. Section V shows experimental results. Finally, in Section VI, we give our conclusions and provide directions for future work.

II. Related work

This section presents prior work on TCP/IP header compression. In particular, we briefly describe three existing TCP header compression algorithms and criticize their use for LLNs. A more detailed discussion of these algorithms can be found in [11].

One of the first TCP/IP header compression methods was Compressed TCP (CTCP), which has been proposed by Jacobson [12]. Jacobson's header compression algorithm distinguishes dynamic fields from static fields. The static fields (e.g., source address, source port, etc) are sent in two situations: when initiating a connection, and when refreshing the context after a loss of synchronization. CTCP proposes to send the difference between the current and the previous value of dynamic fields (e.g., sequence number, acknowledgment number). When the synchronization is lost between the compressor and the decompressor, the TCP sender sends a segment with a regular header to refresh the context. Experimental studies [13]–[15] have shown that the performance of Jacobson's algorithm may degrade significantly in noisy/lossy network environments. An important disadvantage of CTCP is that it does not support TCP options, some of which are ubiquitous nowadays.

IPHC [16] enhances Jacobson's TCP header compression by introducing a mechanism, called TWICE, to repair incorrectly-decompressed headers. TWICE is most efficient when applied to data-carrying TCP segments. [16] also describes a mechanism for explicitly requesting the transmission of less-compressed or uncompressed headers. Such a mechanism is especially suited for pure TCP acknowledgments. Note however that IPHC does not actually provide a compression method for TCP options; changing option fields are carried in compressed headers, but without any compression. Also, the header request mechanism may be unsuited for lossy 6LoWPAN networks, whose low bit rates and strong energy constraints are at odds with any additional signaling overhead. In addition, these two schemes are based on sending a delta for the evolution of the sequence and acknowledgment fields. In case of de-sequencing, this latter leads to a complex reordering. TCPHC enhances IPHC by defining an adapted header compression of TCP for LLNs by sending less significant bytes instead of sending a delta value. Moreover, TCPHC completes IPHC by defining header compression schemes for the mostly used TCP options.

ROHC-TCP [11] improves [16] by providing a new method for compressing all TCP header fields, including the TCP options. ROHC-TCP proposes also to start compressing packets from the SYN segments, using parameters from previous or simultaneous connections. This may offer noticeable improvements in performance when most TCP flows are short-lived, i.e., composed of a small number of data segments. Nevertheless, the ROHC-TCP algorithm is fairly complex and its memory requirements may not be met by small, constrained devices.

The TCPHC algorithm, described in this paper, supports features like the compression of TCP options adapted for LLNs, segments reordering, and at the same time it is relatively simple and easy to implement in memory- and CPU-constrained devices.

III. TCP Header Compression

This section presents LOWPAN_TCPHC (TCPHC), the TCP header compression mechanism for 6LoWPANs. The main purpose of TCPHC is to reduce the protocol header overhead, with the intent of reducing both bandwidth usage and energy consumption due to packet transmissions.

Indeed, TCPHC does not only introduce a header compression algorithm but also provides a scheme to allow establishing TCP connections between an external IP host and a LoWPAN host, and between two LoWPAN hosts. This former type of connection is performed by an Edge Router (ER) which links the 6LoWPAN to an external IPv6-based network. Fig. 1 shows a typical 6LoWPAN topology with three edge routers, which create a bridge between the LoWPAN network and the external IP network. The path between a LoWPAN node to the external network may be changed following the movement of the node or the updating of routing tables.

The compression and the decompression mechanisms are implemented on the edge routers and on the LoWPAN hosts. The external IP host sends and receives regular TCP segments (i.e., with normal TCP header), whereas the LoWPAN host sends and receives segments with compressed headers or full headers (i.e., a regular header with a context identifier). The TCP connection can also be established between two LoWPAN hosts inside the same LLN for Machine-to-Machine (M2M) communications purposes.

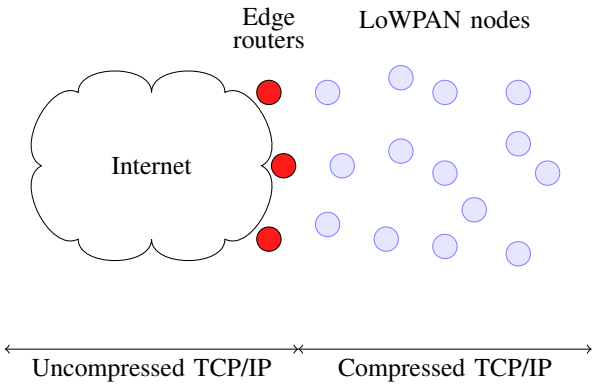


Fig. 1. The IPv6 over Low power Wireless Personal Area Network Topology

Thus, all segments can be either full header segments or compressed header segments. The TCP header compression algorithm defines two kind of headers. For the TCP opening phase, error management messages, or messages with URG pointer set to 1, full header segments are sent. For the other situations, compressed header segments are sent. All of these segments contain the Context Identifier (CID), which is used, with the IPv6 address of the first involved LoWPAN node, to identify the connection during the transfer phase and this way avoids to send on each packet the port numbers. The CID value and its size are assigned by the first LoWPAN node involved in the TCP connection. Figures 2 show the exchanged TCP control segments in the TCP connection establishment phase.

A. Dynamic fields compression

In this section, we define the TCPHC specifications for TCP header compression for IEEE 802.15.4 networks. TCPHC initiates the compression algorithm by exchanging a context identifier at the beginning of the connection. The compressor and decompressor store most fields of the first full headers as a context. The context consists of the header fields whose values are constant. These fields should be elided because they are the same or few change with respect to the previous header. It is more efficient to send fewer bits which are the difference from previous value compared to the sending of the absolute value. This mechanism is based on sending not all TCP fields, but only fields or parts of fields that do change from the last one sent. For example, the most significant bytes of the sequence number field can be elided if they are equal to that of the previous segment. The following paragraphs detail how the TCP dynamic filed are compressed.

- **Sequence and acknowledgement numbers:** the sequence number is the number of first data byte in the segment (except the first segment). The length of the sequence number field is four bytes. In a TCP connection, the sequence number is incremented for each segment by a value which is between 0 and the Maximum Segment Size (MSS). Thus, the two least significant bytes would change more frequently than the two most significant ones. For example, only the least significant byte should be sent if the other bytes do not change. The decompressor module can deduce the elided bytes from the previously received segments. The sequence number can be elided if a receiver does not send data to the source and is acknowledging data segments. The same algorithm is used for the compression of acknowledgement number and only bytes which are changed should be carried inline. If the TCP sink does not generate data, the four bytes of sequence number are omitted in all acknowledgement segments and only compressed acknowledgement fields should be sent.
- **The window** field can be omitted if it does not change in time. The decompression deduces the value of this field from the last received full segment.
- **Flags** are omitted because TCP control messages are sent uncompressed. The not compressed flags are: Push, Urgent, Congestion Window Reduced (CWR) and ECN-Echo indicates (ECE).
- **The urgent pointer** field is sent in uncompressed format only if the urgent flag is set. Otherwise, this field is elided.
- **The checksum** is not reduced and it is used by the receiver to check if the decompressed TCP segment is received correctly.

As a result, TCPHC can compress the TCP header down to five bytes (2-bytes LOWPAN_TCPHC, 1 byte CID, 2-bytes Checksum) if no TCP options are included.

B. Segment loss management

Here, we present how the TCPHC mechanism should react when a segment is lost or is assumed to be lost. The loss is handled when the TCP ACK segment is not received within the round-trip time (RTO). The ER handles a retransmission by scanning the sequence numbers. The ER should send a retransmitted segment without compressing the dynamic fields. This mechanism allows updating the context on both sides after a packet loss. We assume that the 6LoWPAN has a low bit rate, and also that nodes are memory-constrained and thus the TCP window size is probably limited to a few segments. In this case, the loss of synchronization will likely not lead to a burst of losses. For this reason, this document does not present a refresh algorithm to update the context between the compressor and the decompressor.

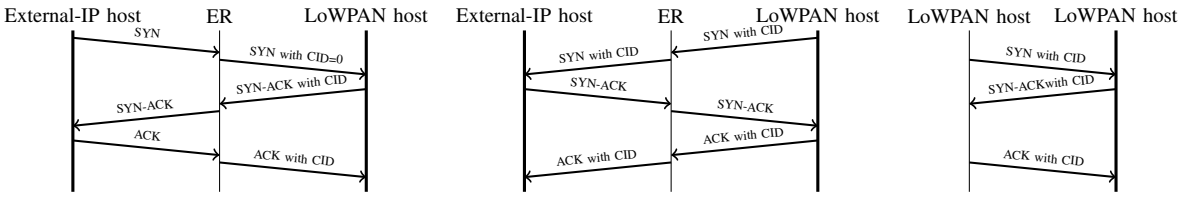
IV. Experimental Setup

In this section, we describe our wireless testbed and the placement of our wireless devices.

A. Physical Setup

In our wireless testbed, seven wireless nodes are distributed with the same distance (between three and four meters) between each neighbor. The position of the wireless device is shown in Fig. 3. We can distinguish four types of wireless devices based on their functionalities:

1) The Edge Router (ER) is the border router that connects the wireless network to the IP-based wired network.



(a) TCP connection initiated by an external IP-host (b) TCP connection initiated by a LoWPAN host (c) TCP connection between two LoWPAN hosts

Fig. 2. TCP connection initiation in 6LoWPAN

2) Wireless nodes (N1, N2, N3, N4, and N5) can either be a Terminal Node (TN), or only a relay of data frames from the ER to the TN and vice-versa.
3) The External Node (EN) is an external sensor node in the same wireless network, which generates a CBR traffic to increase the packet loss ratio.

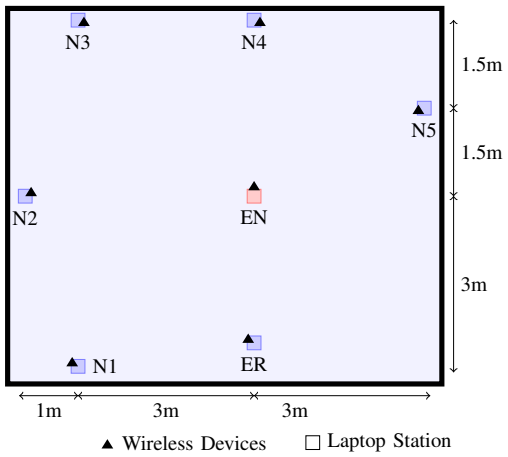


Fig. 3. The distribution of the wireless motes in the testbed

B. Hardware setup

In our testbed, all wireless devices are connected to a standard laptop by the USB port. This solution allows us to log the output messages from the wireless devices to the standard laptop. The embedded device used in our testbed is Crossbow TelosB [4] mote. It uses Texas Instruments MSP430 microcontroller, which offers a 10kB RAM, and a 48 kB program flash memory. The Crossbow TelosB radio is CC2420 [5], which uses ISM frequency band (from 2400 MHz to 2483.5 MHz) and offers 250 kbps data rate. Before starting our experiments, we have explored the use of radio channels. We found that the radio environment is highly polluted by the IEEE 802.11 networks, because the IEEE 802.15.4 uses the radio frequency which are the same as that of IEEE 802.11. To reduce this effect, we used the last channel of IEEE 802.15.4 (channel number 26) as described in [8].

[4]http://www.hoskin.qc.ca/uploadpdf/Instrumentation/CrossBow/hoskin_TPR2400CA_42efb73715b8b.pdf

[5]www.flexipanel.com/Docs/CC2420_Data_Sheet_1_2.pdf

C. Software setup

In our work, we use Contiki OS as the operating system for our wireless devices. We have chosen Contiki OS because it already implements 6LoWPAN, UDP, and TCP. Contiki OS [6] is open source operating system for networked embedded devices that includes the uIPv6 [17] stack. Moreover, Contiki OS provides standard operating system features like threads, timers, random number generator, clocks, a file system, and a command line shell.

The 6LoWPAN implementation in Contiki OS is conformant to to [5].TCP is partially implemented on Contiki OS because of the memory-constraint of the wireless devices. In uIP, all RFC requirements that affect host-to-host communication are implemented, except for some mechanisms, such as soft error reporting. A more detailed description of TCP implementation on Contiki OS is as follows:

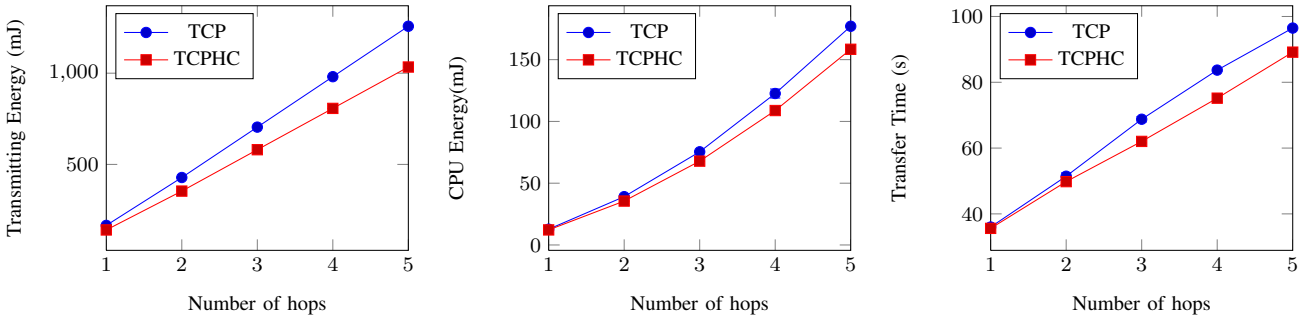
- Maximum Segment Size (MSS): the default value of MSS in Contiki is 48 bytes. Fragmentation and recovery is not implemented, instead, a short MSS is utilised to send a TCP segment in one IEEE 802.15.4 frame.
- Retransmissions: driven by the periodic TCP timer. Since there is not any buffer to remember the packets sent previously, Contiki OS requires that the application takes an active part in performing the retransmissions.
- Flow Control: in uIPv6, the application cannot send more data than that the receiving host can buffer. In addition, the application cannot send more data than the amount of bytes it is allowed by the receiving host.
- Congestion Control: there is no congestion control mechanism implemented on Contiki OS because uIPv6 can handle only one in-flight TCP segment per connection.

D. Energy Consumption

Contiki OS provides a tool to compute the running time spent by a node on the *transmit* and *listen* radio states. Moreover, the Contiki OS provides an estimation of its CPU consumption. Table I shows the energy power values, that have been obtained from CC2420 Datasheet and the Texas Instruments MSP430 Datasheet.Based on those values, we are able to compute the consumed energy. For example, the consumed energy E_{Listen} due to channel listening is equal to

$$E_{\text{Listen}} = T_{\text{Listen}} \times \text{Voltage} \times I_{\text{Listen}} \quad (1)$$

[6]www.sics.se/contiki



(a) Transmit Energy (b) CPU Energy (c) Transfer Time

Fig. 4. Experimental results of multi-hop TCP vs. TCPHC over 6LoWPAN end-to-end retransmissions times without concurrent CBR traffic.

where T_{Listen} and I_{Listen} are respectively the time spent by a mote in listen mode and the listen current.

TABLE I
ENERGY PARAMETERS

Parameter	Value
Voltage	3 V
CPU	0.5 mA
Transmit	17.4 mA
Listen	19.7 mA

V. RESULTS AND DISCUSSION

In this section, we describe the TCP scenario in details. A TCP source sends 48 kbytes of data to a TCP receiver. The maximum segment size is equal to 48 bytes, thus the TCP source sends 1000 TCP data segments to the receiver. We design two kinds of experiments: loss-free scenario and lossy scenario. In our scenario, one node is a TCP source and the other are relay nodes. With respect to Figure 3, the routing is such that for a i-hops scenario, Ni is the destination node and nodes N1 to N$i-1$ (in that order) relay TCP segments from ER to Ni.

In the experiment, we are mainly interested in the energy consumed by the CPU, the radio transmission, and the radio listening during the TCP connection. We are also interested in the number of segment retransmissions and throughput. We study the TCP performance in a multi-hop scenario from one to five hops in terms of both throughput and energy consumption. In our experiment, there are two traffics, one is a TCP connection, the other is a constant bit rate (CBR) traffic generated by a UDP traffic to increase the contention and to increase the bit error rate.

A. TCPHC performance in free-losses environment

In a first scenario, we compare legacy TCP to TCP with header compression, in terms of transmitting energy consumption and transfer time for different numbers of hops.

Fig. 4 (a) shows that TCPHC reduces the energy consumption. Firstly, we do not distinguish a significant improvement for less than three hops between the two TCP hosts. However, the TCPHC reduces about 17% of all the energy consumed by all sensor nodes when there are 5 hops between the TCP sender and the TCP receiver. Fig. 4 (b) shows that the header compression algorithm does not increase the consumed CPU energy. On the contrary, the CPU energy has been reduces by TCPHC. In addition, Fig. 4 (c) shows that the transfer time decreases when TCPHC is deployed. TCPHC reduces the transfer duration by about 9 percent compared to legacy TCP.

B. TCPHC performance in Lossy environment

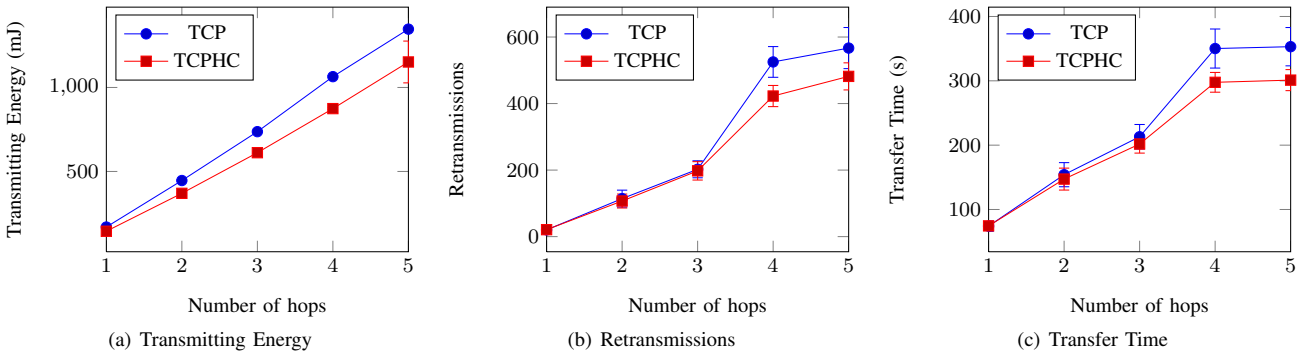
Now, we compare legacy TCP to TCP with header compression in a lossy network. We add a new traffic generated by the External Node with the purpose of increasing the collisions and thus the BER. To compare TCP performance with and without the header compression algorithm, we plot the energy consumption, the TCP end-to-end retransmissions and transfer time with different number of hops between the TCP sender and TCP receiver.

Fig. 5 (a) shows that the transfer time decreases when TCPHC is deployed. The header compression algorithm reduces the transfer duration by about 15 percent compared to legacy TCP. The transfer time is reduced due to the short size of the new segments which require less time to be sent and received. Moreover, Fig. 5 (b) shows that TCPHC reduces the number of TCP end-to-end retransmissions compared to Legacy TCP.

Finally, Fig. 5 (c) shows that the header compression algorithm reduces significantly the total energy consumption. As discussed in Fig. 4, we do not observe a significant improvement when the number of hops between the two TCP hosts is less than or equal to two. However, the TCPHC reduces about 15% of all the energy consumed by all sensor networks when there are 5 hops between the TCP sender and the TCP receiver. All those results show that TCPHC is a very interesting scheme for making TCP more energy-efficient and viable for Low-power and Lossy Networks.

VI. CONCLUSION

This paper has presented experimental results of a TCP header compression algorithm over 6LoWPAN in a multi-hop scenario. Experimental results have shown that the larger



(a) Transmitting Energy (b) Retransmissions (c) Transfer Time

Fig. 5. Experimental results of multi-hop TCP vs. TCPHC over 6LoWPAN end-to-end retransmissions times with a concurrent CBR traffic.

the distance between the TCP sender to the edge router, the more header compression improves energy efficiency. Our experimental study of TCP performance over 6LoWPAN should encourage industrial companies to implement TCPHC in their future wireless devices.

VII. Acknowledgements

The work of Ahmed Ayadi has been funded by the Pôle de Recherche Avancée en Communications (PRACom).

References

[1] N. Ahmed, M. Rutten, T. Bessell, S. Kanhere, N. Gordon, and S. Jha, "Detection and tracking using particle-filter-based wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 9, pp. 1332 –1345, sep. 2010.

[2] R. Jafari, A. Encarnacao, A. Zahoory, F. Dabiri, H. Noshadi, and M. Sarrafzadeh, "Wireless sensor networks for health monitoring," in *Proceedings of Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, jul. 2005, pp. 479 – 481.

[3] M. Kranz, P. Holleis, and A. Schmidt, "Embedded interaction: Interacting with the internet of things," *Internet Computing, IEEE*, vol. 14, no. 2, pp. 46 –53, 2010.

[4] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC 4919, IETF, United States, 2007. [Online]. Available: http://tools.ietf.org/html/rfc4919

[5] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams in 6LoWPAN Networks," Internet Draft draft-ietf-6lowpan-hc-11, work in progress, September 2010. [Online]. Available: http://tools.ietf.org/html/draft-ietf-6lowpan-hc-11

[6] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, IETF, United States, September 2007. [Online]. Available: http://tools.ietf.org/html/rfc4944

[7] J. Postel, "User Datagram Protocol," Internet Standards Track RFC 768, IETF, August 1981.

[8] IEEE Computer Society, "IEEE Std. 802.15.4-2006," October 2006.

[9] E. Stephen, "Internet Protocol, Version 6 (IPv6) Specification," IETF, RFC 2460, December 1998.

[10] A. Ayadi, D. Ros, and L. Toutain, "TCP header compression for 6LoWPAN," Internet Draft draft-aayadi-6lowpan-tcphc-00, work in progress, July 2010. [Online]. Available: http://tools.ietf.org/html/draft-aayadi-6lowpan-tcphc-00

[11] L.-E. Jonsson, G. Pelletier, and K. Sandlund, "The RObust Header Compression (ROHC) Framework," RFC 5795, IETF, United States, July 2007.

[12] V. Jacobson, "Compressing TCP/IP headers for Low-Speed Serial Links," RFC 1144, IETF, Febrary 1990.

[13] S. J. Perkins and M. W. Mutka, "Dependency Removal for Transport Protocol Header Compression over Noisy Channels," in *Proceedings of International Conference on Communications (ICC)*, 1997, pp. 1025–1029.

[14] A. Srivastava, R. Friday, M. Ritter, and W. Filippo, "A study of TCP performance over wireless data networks," in *Proceedings of IEEE VTS 53rd Vehicular Technology Conference*, vol. 3, 2001, pp. 2265 –2269.

[15] R. Wang, "An experimental study of TCP/IP's Van Jacobson header compression behavior in lossy space environment," in *Proceedings of 60th IEEE Vehicular Technology Conference*, vol. 6, sep. 2004, pp. 4046 – 4050.

[16] M. Degermark, B. Nordgren, and S. Pink, "IP Header Compression," IETF, RFC 2507, United States, February 1999.

[17] A. Dunkels, "Full TCP/IP for 8 Bit Architectures," in *Proceedings of First ACM/Usenix International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*, San Francisco, May 2003. [Online]. Available: http://www.sics.se/~adam/mobisys2003.pdf