Network Working Group	A. Ayadi	
Internet-Draft	D. Ros	
Intended status: Experimental	L. Toutain	
Expires: April 28, 2011	Telecom Bretagne	
	October 25, 2010	

TOC

TCP header compression for 6LoWPAN
draft-aayadi-6lowpan-tcphc-01

Abstract

This document describes LOWPAN_TCPHC, a scheme for compressing the header of Transmission Control Protocol (TCP) segments, in order to reduce the overhead on low-power and lossy networks. It also specifies the LOWPAN_TCPHC header fields for the transmission of TCP segments over IPv6 for Low-power Wireless Personal Area Networks (6LoWPAN). In many cases, the 20 bytes of the mandatory TCP header can be compressed into as little as 6 bytes.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”
This Internet-Draft will expire on April 28, 2011.

Copyright Notice

Table of Contents

1. Introduction

TOC

The 6LoWPAN Working Group [RFC4919] (Kushalnagar, N., Montenegro, G., and C. Schumacher, “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals,” August 2007.) has proposed LOWPAN_IPHC [I-D.ietf-6lowpan-hc] (Hui, J. and P. Thubert, “Compression Format for IPv6 Datagrams in 6LoWPAN Networks,” October 2009.), a new version of the LOWPAN_HC1 header compression mechanism [RFC4944] (Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” September 2007.) which reduces the IPv6 header to about 3-5

bytes. In [I-D.ietf-6lowpan-hc] (Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams in 6LoWPAN Networks," October 2009.), a header-compression method for the transport layer (LOWPAN_NHC) has also been introduced, however, only a UDP [RFC0768] (Postel, J., "User Datagram Protocol," August 1980.) datagram compression mechanism is specified.

UDP header compression is useful for 6LoWPAN because many Low-power and Lossy Network (LLN) applications are fault-tolerant and do not require 100% reliability. However, other applications and services such as SSH and HTTP require a reliable service from the transport layer that cannot be offered by UDP. Moreover, some usage scenarios of LLNs, such as health, military and security applications, impose strong reliability constraints. Also, in some usage cases (e.g., sending a software update to a wireless node, or sending a query requesting a specific information from the wireless node) there is a need for a reliable data transport.

In this document, we focus on the Transmission Control Protocol (TCP) [RFC0793] (Postel, J., "Transmission Control Protocol," September 1981.), which carries most of the traffic in IP networks. TCP is a connection-oriented, end-to-end reliable transport protocol. To ensure end-to-end reliability, TCP must recover from packet corruption, loss or out-of-order delivery. This is achieved by assigning a sequence number to each transmitted byte (done by the TCP source), by requiring a positive acknowledgment (ACK) from the TCP destination, and by retransmitting lost or corrupted packets.

In the context of 6LoWPAN networks, the size of TCP headers may induce a large overhead, especially for link-layer technologies that use small frames. For instance, 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 typically [IEEE 802.15.4] (IEEE Computer Society, "IEEE Std. 802.15.4-2006," October 2006.) MAC frame. In addition, 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 [RFC1122] (Braden, R., "Requirements for Internet Hosts - Communication Layers," October 1989.) is not used). This suggests that the use of header-compression mechanisms for TCP may result in important performance gains, in terms of used bandwidth and/or energy spent for frame transmission.

A TCP header compression algorithm for 6LoWPAN should respect some requirements: (a) Efficiency (the scheme must provide low overhead in all cases), (b) Transparency (the resulting header after a compression and decompression should be identical to the original header), (c) Reordering tolerance (the scheme should be able to decompress compressed segments correctly even when segments arrive with a moderate reordering).

The goal of this document is thus to define a TCP header compression scheme for 6LoWPAN, called LOWPAN_TCPHC, which allows to significantly reduce the TCP overhead. The TCP header compression and decompression is performed in the edge router between the 6LoWPAN and the external IP

network. The compression scheme can also be used between two 6LoWPAN nodes for machine-to-machine communications. This document defines also an encoding format for LOWPAN_TCPHC header compression. Such mechanism and packet format aims at making TCP a more viable proposition for 6LoWPAN networks. Moreover, the LOWPAN_TCPHC mechanism can be used with LOWPAN_IPHC [I-D.ietf-6lowpan-hc] (Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams in 6LoWPAN Networks," October 2009.) and thus reduce all header overheads to about seven to ten bytes instead of 60 bytes.
The proposed scheme does not compress TCP control messages at the connection establishment phase. Those TCP segments are used to exchange a context identifier. Such context identifier replaces the port numbers in subsequent TCP segments, as a means of identifying a given TCP session. Some TCP options, such as Timestamp [RFC2018] (Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options," October 1996.) and SACK [RFC1323] (Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance," May 1992.) [RFC2883] (Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, "An Extension to the Selective Acknowledgment (SACK) Option for TCP," July 2000.), are supported (i.e., compressed) by the mechanism, while other options, unlikely to be required/used in 6LoWPANs, are omitted.

1.1. Related Work

TOC

This section presents prior work on TCP/IP header compression. In particular, we will briefly describe three existing TCP header compression algorithms. A more detailed discussion of these algorithms can be found in [RFC4996] (Pelletier, G., Sandlund, K., Jonsson, L-E., and M. West, "RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP)," July 2007.).
One of the first TCP/IP header compression methods was Compressed TCP (CTCP), proposed by Jacobson [RFC1144] (Jacobson, V., "Compressing TCP/IP headers for low-speed serial links," February 1990.). Jacobson's header compression algorithm distinguishes between dynamic fields and static fields. The static fields (e.g., source address, source port, ...) 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 [Perkins et al.] (Perkins, S. and M. Multa, "Dependency removal for transport protocol header compression over noisy channels," June 1997.) [Srivastava et al.] (Srivastava, A., Friday, R., Ritter, M., and W. San Filippo, "A study of TCP performance over wireless data networks," May 2001.) [Wang04] (Wang, R., "An

experimental study of TCP/IP's Van Jacobson header compression behavior in lossy space environment," September 2004.) 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 [Medina05] (Medina, A., Allman, M., and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet," April 2005.).
IPHC [RFC2507] (Degermark, M., Nordgren, B., and S. Pink, "IP Header Compression," February 1999.) 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. [RFC2507] (Degermark, M., Nordgren, B., and S. Pink, "IP Header Compression," February 1999.) also describes a mechanism for explicitly requesting the transmission of less-compressed or uncompressed headers. such 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, which low bit rates and strong energy constraints are at odds with any additional signaling overhead. LoWPAN_TCPHC enhances IPHC by defining an adapted header compression of TCP for LLNs by sending lest significant bytes instead of sending a delta value. Moreover, LoWPAN_TCPHC completes IPHC by defining header compression schemes for the mostly used TCP options.
ROHC-TCP [RFC4996] (Pelletier, G., Sandlund, K., Jonsson, L-E., and M. West, "RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP)," July 2007.) improves on [RFC2507] (Degermark, M., Nordgren, B., and S. Pink, "IP Header Compression," February 1999.) by providing a new method for compressing all TCP header fields, including the TCP options. ROHC-TCP proposes also to start compressing packets starting 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 algorithm described in this document, LOWPAN_TCPHC, supports features like the compression of TCP options and FIN segments, and at the same time it is relatively simple and easy to implement in memory- and CPU-constrained devices.

1.2. Terminology

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] (Bradner, S.,

"Key words for use in RFCs to Indicate Requirement Levels," March 1997.).

This section defines general terms related to TCP/IPv6 header compression [RFC2507] (Degermark, M., Nordgren, B., and S. Pink, "IP Header Compression," February 1999.) [RFC4995] (Jonsson, L-E., Pelletier, G., and K. Sandlund, "The RObust Header Compression (ROHC) Framework," July 2007.) and to the 6LoWPAN architecture [RFC4919] (Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," August 2007.) used in this specification.

*Subheader: An IPv6 header, a UDP header, or a TCP header.

*Header: A chain of subheaders.

*Compression: The act of reducing the size of a header by either removing or reducing the size of header fields. This is done in a way such that a decompressor can reconstruct the header if its context state is identical to the context state used when compressing the header.

*Decompression: The act of reconstructing a compressed header.

*Static fields: These fields are expected to be constant throughout the lifetime of the TCP connection. Static information must be communicated once in some way.

*Dynamic fields: These fields are expected to vary between different TCP segments, either randomly within a limited range or in some other manner.

*Context identifier (CID): A small unique number identifying the context that should be used to decompress a compressed header. Carried in full headers and compressed headers.

*Context: The state used by the compressor to compress a header, and by the decompressor to decompress a header. The context is given by the uncompressed version of the last header sent (compressor) or received (decompressor) over the link, except for fields in the header that are included "as-is" in compressed headers, or that can be inferred from e.g. the size of the link-layer frame.

*Full header: An uncompressed header that updates or refreshes the context for a packet stream. It carries a CID that will be used to identify the context.

*Regular header: A normal, uncompressed header. It does not carry any CID.

*Compressed header: A header in which all the static fields are elided, and all the dynamic fields are sent compressed.

*Incorrect decompression: When a decompressed header does not match the corresponding original, uncompressed header. Usually due to mismatching contexts between the compressor and decompressor, caused by e.g. bit errors during the transmission of the compressed header, or by packet loss.

*IEEE 802.15.4: A low-power, low-bandwidth link layer protocol.

*LoWPAN host: A node that only sources or sinks IPv6 datagrams. Referred to as a host in this document.

*LoWPAN router: A node that forwards datagrams between arbitrary source-destination pairs using a single 6LoWPAN interface, performing IP routing on that interface.

*LoWPAN edge router (ER): An IPv6 router that interconnects the 6LoWPAN to another IP network. Referred to as an Edge Router in this document.

*LoWPAN node: A node that composes a 6LoWPAN, referring to both hosts and routers. Simply called a Node in this document.

---

2. Protocol Overview

TOC

This section gives an overview of the TCP header compression mechanism for 6LoWPAN (LOWPAN_TCPHC). The main purpose of LOWPAN_TCPHC is to reduce the protocol header overhead, with the intent of reducing both bandwidth usage and energy consumption due to packet transmissions. Indeed, the LOWPAN_TCPHC allows establishing TCP connections between an external IP host and a LoWPAN host, and also 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. Figure 1 (A 6LoWPAN network) shows a typical 6LoWPAN topology with which two edge router create a bridge between the LoWPAN network and the external IP network. The path between a LoWPAN node to the external network may change following the movement of the node or the update of routing tables.

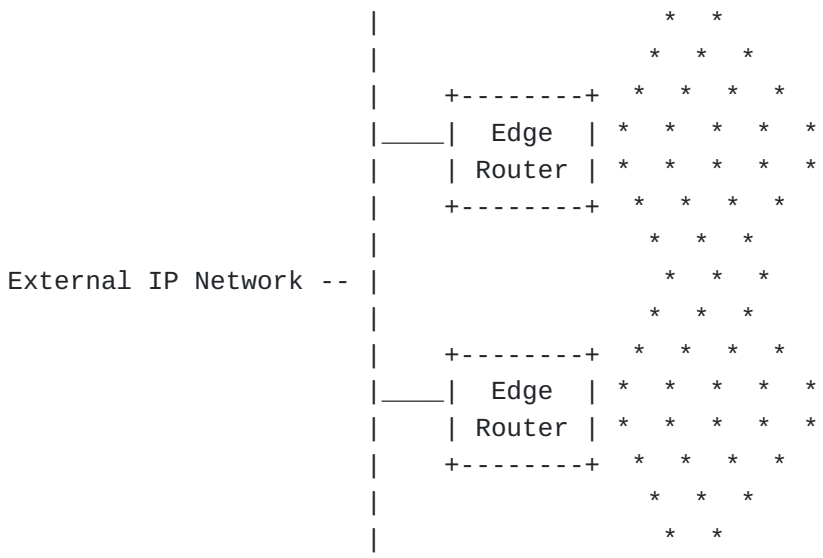


Figure 1: A 6LoWPAN network

The compression and decompression mechanisms are implemented on the edge routers and on the LoWPAN hosts. The ERs create and maintain the contexts of all TCP connections. Figure 2 (TCP Connection initiated by an external IP-host) and Figure 3 (TCP Connection initiated by a LoWPAN host) show sequence diagrams of a connection establishment between a LoWPAN host and an external IP host. The external IP host sends and receives regular TCP segments, whereas the LoWPAN host sends and receives segments with compressed headers or full headers.

The LOWPAN_TCPHC algorithm does not compress TCP control messages at the connection establishment phase. These segments are used to exchange the context identifier (CID) and allow the ER to create a context using the TCP header fields. The LOWPAN_TCPHC algorithm supports usefull TCP option for LLNs. Supported TCP options that are negotiated in the two first messages are sent in a full header format. Whereas, the remaining supported options are sent compressed with in the TCP segments. The LOWPAN_TCPHC algorithm defines a compression mechanism for TCP SACK and Timestamp options.

The TCP connection may also be established between two LoWPAN hosts for Machine-to-Machine communications. In this case, the context should be shared between the two LoWPAN hosts. If a packet is dropped due to loss, then the mechanism refreshes the context by retransmitting lost segments using the mostly compressed header format.

The compression protocol uses two different header formats. For the TCP opening phase and for error management, the TCP segments must be sent with a full header. These segments contain the Context Identifier (CID) which will be used to identify the connection during the transfer phase. The CID replaces the two port numbers, hence avoiding to send

the port numbers in every packet. The CID value and its size are set by the LoWPAN host if the TCP connection is between a LoWPAN host and an external IP host. Otherwise (i.e., a TCP connection between two LoWPAN hosts), the CID value and its size are set by the LoWPAN host that has opened the connection. For the TCP connection between a LoWPAN host and an external IP host, the couple (CID, Ipv6 address) must be unique. The second kind of header is the compressed header in which all static fields are elided, and all dynamic fields are compressed. Depending on how they change with respect to the last sent segment, dynamic fields may be compressed fully (i.e., elided) or partially (i.e., only a portion of a field is sent, containing the bytes that have changed). The last kind of packet is sent when a packet is lost, corruption or reordering is detected by the TCP receiver. This segment is called mostly compressed header and contains dynamic fields with all bytes uncompressed, while its static fields are elided. This kind of segment should be sent after a loss of synchronization between the compressor and the decompressor.

2.1. Connection Initiation

TOC

This section gives an overview of TCP connection initiation with LOWPAN_TCPHC.

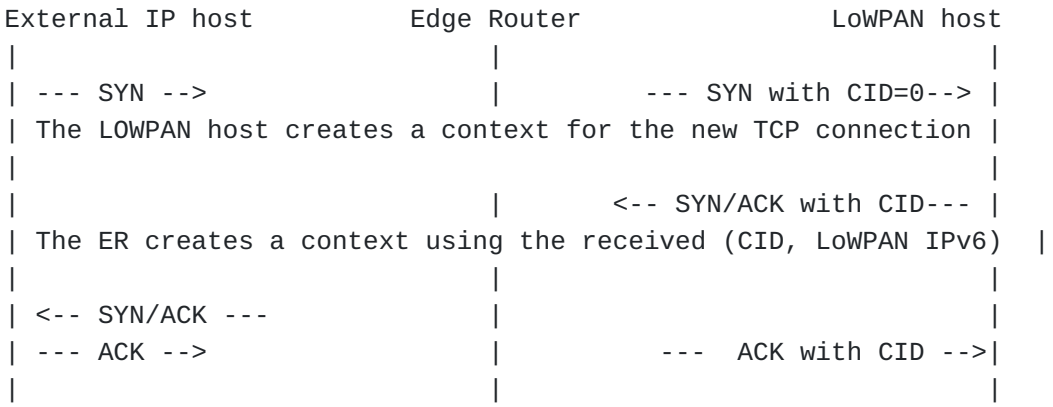


Figure 2: TCP Connection initiated by an external IP-host

Figure 2 (TCP Connection initiated by an external IP-host) shows an example of a connection initiation scenario started by an external IP host. In this scenario, an external host sends a SYN segment trying to establish a connection with a LoWPAN host. The SYN message can include options, some of which may be eliminated by the Edge Router (i.e.,

those options not supported by the LOWPAN_TCPHC mechanism). The ER sends the SYN segment in a full header message with a CID equal to zero. Upon the reception of the SYN segment, the LoWPAN node sends a SYN/ACK segment including a new CID, set to the smallest available CID value. The ER creates a new context with the received information from the SYN/ACK segment.

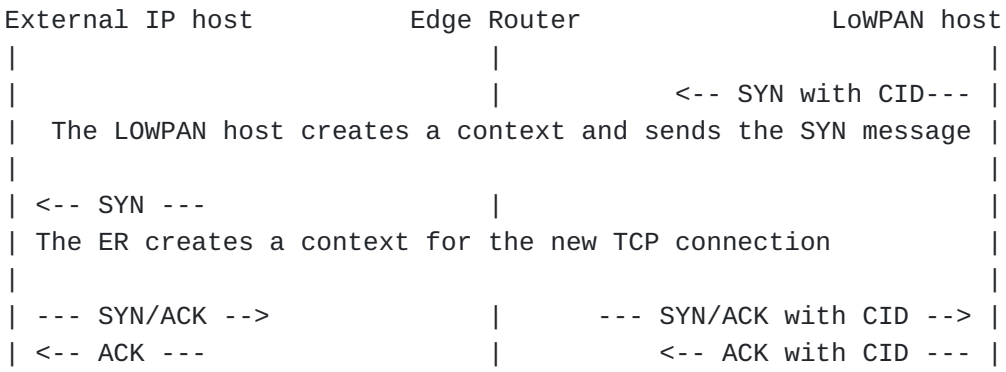


Figure 3: TCP Connection initiated by a LoWPAN host

Figure 3 (TCP Connection initiated by a LoWPAN host) gives an example of a TCP connection initiated by a LoWPAN host. The first SYN segment is sent with the full header with a CID value equal to smallest available value of CIDs. Contrary to the first case, the ER creates a context upon the reception of the SYN message from the LoWPAN host.

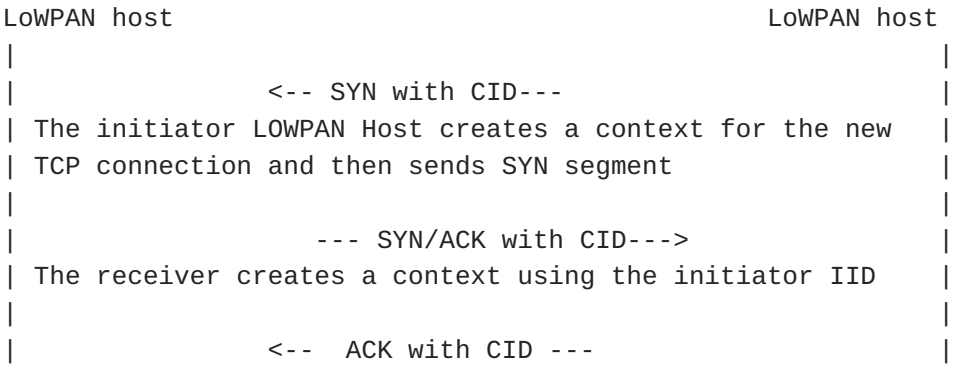


Figure 4: TCP/IPv6 Connection between two LoWPAN hosts

Figure 4 (TCP/IPv6 Connection between two LoWPAN hosts) presents a TCP connection initiated by a LoWPAN node to another LoWPAN node. The LoWPAN host sends in the TCP SYN segment with a CID equals to the smallest available CID value.
The CID value is chosen by the first LoWPAN host that is involved in establishing the TCP connection. This way, it is easy to ensure the unicity of the CID. At the same time, this simplifies the context management at the ERs. Moreover, if all CID values are used, the initiator should increase the CID field length from 8 bits to 16 bits.

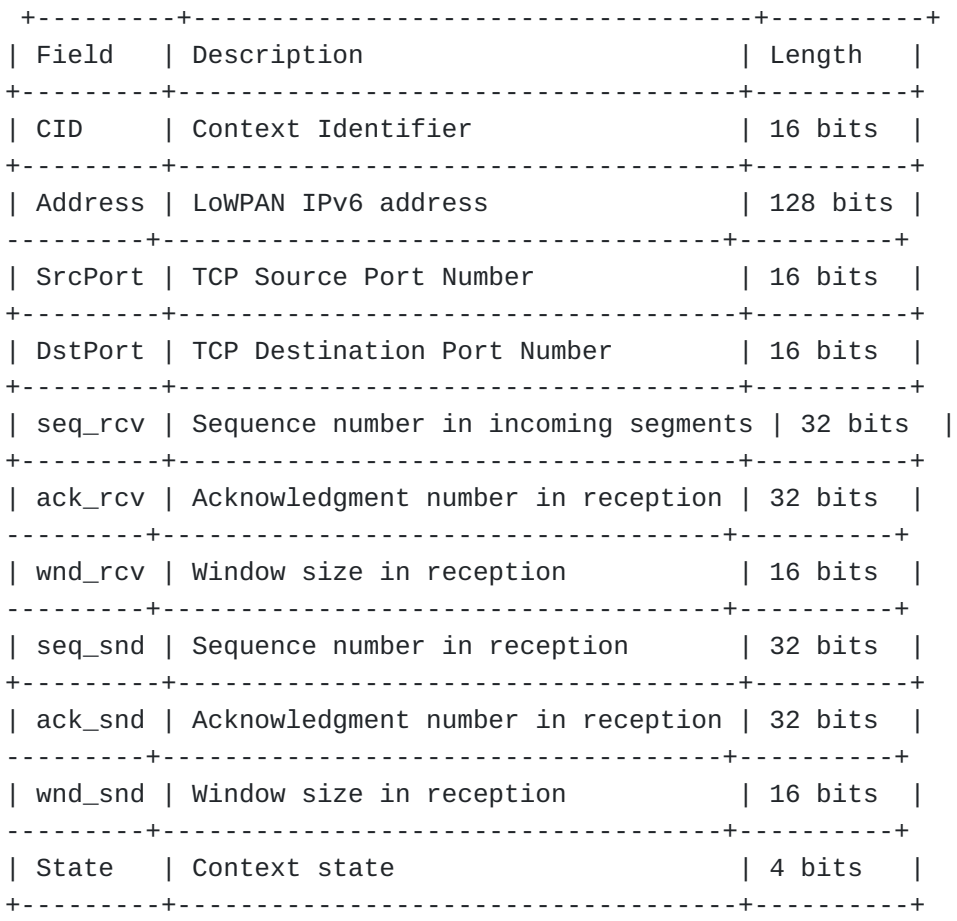
2.2. The LoWPAN_TCPHC context

TOC

2.2.1. LoWPAN_TCPHC context structure

TOC

The context includes some TCP header fields that are needed by the compressor and the decompressor algorithm. Figure 5 (Structure of a LOWPAN_TCPHC context) lists the contents of a LOWPAN_TCPHC context.



Field	Description	Length
CID	Context Identifier	16 bits
Address	LoWPAN IPv6 address	128 bits
SrcPort	TCP Source Port Number	16 bits
DstPort	TCP Destination Port Number	16 bits
seq_rcv	Sequence number in incoming segments	32 bits
ack_rcv	Acknowledgment number in reception	32 bits
wnd_rcv	Window size in reception	16 bits
seq_snd	Sequence number in reception	32 bits
ack_snd	Acknowledgment number in reception	32 bits
wnd_snd	Window size in reception	16 bits
State	Context state	4 bits

Figure 5: Structure of a LOWPAN_TCPHC context

The address field saves the Ipv6 address of the 6LoWPAN node. If the TCP connection is established between two 6LoWPAN nodes, the Ipv6 address of the initiator is saved in the context.
State field indicates in which state a TCP connection is. This field is especially needed by the ERs. The possible states of a context are: closed, using, closing, fin_1, fin_2, fin_3, and shutting.
The CID and Ipv6 address are utilized to identify the connection for compressed headers. The SrcPort and DstPort, toghether with the Ipv6 adsress, are used to identify the connection for full headers.
The seq_rcv, ack_rcv, and wnd_rcv are used to store the dynamic fields of the last incoming segment (except retransmitted segment). While the seq_snd, ack_snd, and wnd_snd are the dynamic fields of the last outgoing segment (except retransmitted segment).

TOC

2.2.2. Context management

This section describes the context management in 6LoWPAN when LOWPAN_TCPHC is used. The edge router to which a LoWPAN node host is attached may change over time, due to route instability or to host mobility However, this change should not break the TCP communication. To ensure the TCP communication despite the change of ER, the ERs should share the contexts of current connections. So, even if a 6LoWPAN node changes its attached ER, the new ER should continue to compress the segments using the same context. Context exchange and management between ERs is out of the scope of this document.
The edge router should free a context when a TCP connection is finished (e.g., reception of FIN control messages). The Edge Router can also free a connection after a silent period (i.e., when no messages are exchanged after a certain period of time).
The ER may remove the context of a TCP connection that is not yet closed. In this case, after receiving a new data segment, the ER SHOULD reply by sending a RST segment to the sender.

2.3. Loss detection and retransmissions

TOC

In this section, we present how the LOWPAN_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 RTO. The ER handles a retransmission by scanning the sequence numbers. The ER should send a mostly compressed header segment when it receives an already sent segment. 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.

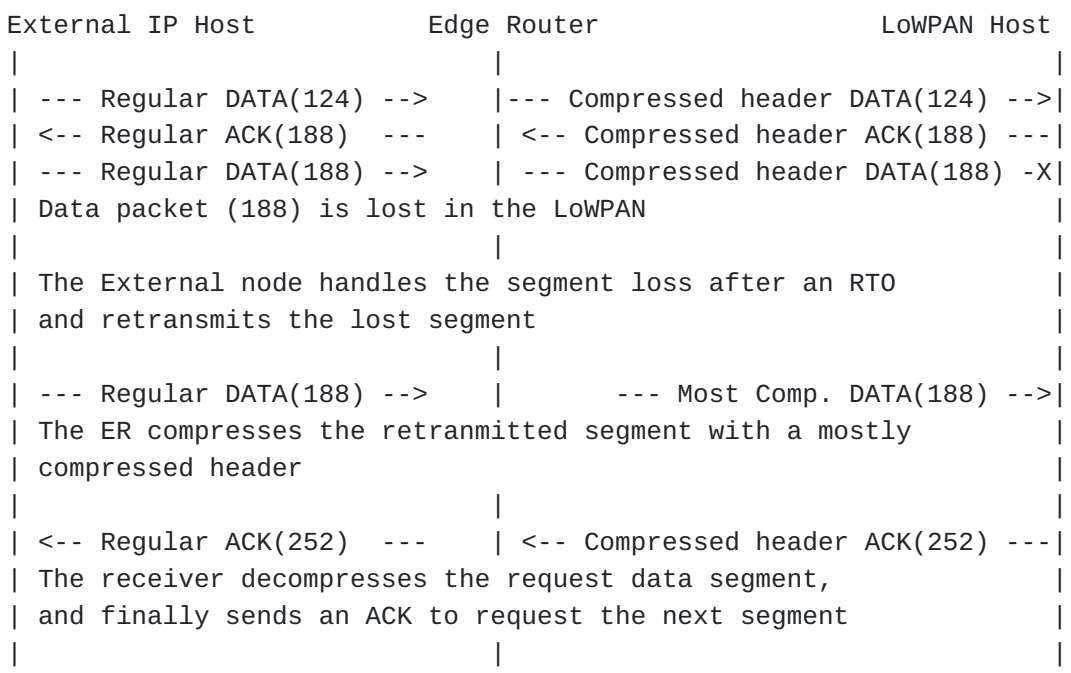


Figure 6: Loss detection in 6LoWPAN

Figure 6 (Loss detection in 6LoWPAN) shows a scenario where a TCP segment is lost in the 6LoWPAN. After an RTO the enternal IP host retransmits the lost segment. Upon the reception of the retransmitted segment, the ER produces a mostly compressed header allowing the LoWPAN node to decompress the segment.

---

3. Transmission Control Protocol

TOC

This section presents more details on TCP and its header fields. As it has been defined in [RFC0793] (Postel, J., “Transmission Control Protocol,” September 1981.), The TCP is a connection-oriented, end-to-end reliable transport protocol mostly used in IP-based networks. The TCP is able to transfer a continuous stream of bytes in each direction between two end-points by packing some number of bytes into segments for transmission through IP-based network.
To ensure the end-to-end reliability, the TCP must recover data that is damaged, lost or delivered out-of-order. This is achieved by assigning a sequence number to each transmitted byte (done by the TCP source), and by requiring a positive acknowledgment (ACK) from the TCP destination. If the ACK is not received within the timeout interval, the data segment is assumed to be lost. Then, the source TCP should retransmit it. At the receiver, the sequence numbers are used to

correctly order segments that may be received out of order and eliminate duplicates. Damaged segments are handled by adding a checksum to each segment transmitted, checking it at the receiver and rejecting damaged segments.

3.1. TCP headers fields

TOC

In this section, we present a short description of TCP header fields and how they are handled by the compression mechanism. [RFC4413] (West, M. and S. McCann, “TCP/IP Field Behavior,” March 2006.) provides a detailed description of TCP header and TCP header options.

*Source port (16 bits): This field identifies the sending port. This field will be replaced by the CID in compressed headers.

*Destination port (16 bits): This field identifies the receiving port. This field will be replaced by the CID in compressed headers.

*Sequence Number (32 bits): The sequence number of the first data byte in this segment (except when SYN flag is set). If SYN is present the sequence number is the initial sequence number (ISN) and the first data byte is ISN+1. Only the bytes that change (respect to the previous segment) will be sent. If this field does not change, nothing should be sent.

*Acknowledgment number (32 bits): If the ACK flag is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent. Only changed bytes of this field according to the last sent segment will be sent. If this field does not change, nothing should be sent.

*Reserved (4 bits): Reserved for future use. Must be zero. This field should not be sent in compressed segments.

*Flags (8 bits):

> **URG (1 bit):** Urgent Pointer field contains a valid value.
>
> **ACK (1 bit):** Acknowledgment field contains a valid value.
>
> **PSH (1 bit):** Push.
>
> **RST (1 bit):** Reset the connection.
>
> **SYN (1 bit):** Synchronize sequence numbers.

FIN (1 bit):
No more data from sender.

CWR (1 bit): Congestion window reduced.

ECE (1 bit): Echo the 'congestion experienced' signal in the IP header.

*Window (16 bits): The number of data bytes, beginning with the one indicated in the acknowledgment field, the sender of this segment is willing to accept. This field is compressed and only the bytes that have changed are sent.

*Checksum (16 bits) The 16-bit checksum field is used for error-checking of the header and data. This field is not compressed by LOWPAN_TCPHC.

*Urgent Pointer (16 bits): This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the byte following the urgent data. This field is only interpreted in segments with the URG flag set to 1. This field is rarely used in TCP communications. For this reason, the URG flag and Urgent Pointer are sent if and only if they are set. In this case, the segment is sent with a full header.

3.2. TCP Header Options

TOC

Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. Options are considered for the checksum calculation. The most used TCP options are indicated below:

*End of Option List (8 bits): indicates the end of the option list.

*No-Operation (8 bits): may be used between options.

*Maximum Segment Size (32 bits): the maximum segment size at the TCP which sends this segment.

*Window Scale Option (24 bits): indicates that the sending TCP end-host is prepared to perform both send and receive window scaling.

*SACK-Permitted (16 bits): ask for using SACK option.

*SACK (80-320 bits): selective acknowledgment.

*Timestamp (80 bits): used to compute RTT.

The TCP header padding is used to ensure that the TCP header ends and data begins on the 32-bit boundary.

---

4. TCP header fields compression

TOC

In this section, we define the LOWPAN_TCPHC specifications for TCP header compression for LLNs. LOWPAN_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.

As described in [RFC2507] (Degermark, M., Nordgren, B., and S. Pink, “IP Header Compression,” February 1999.), the context consists of the header fields whose values are constant or regularly increasing. The dynamic fields should be elided because they are the same with respect to the previous header. In fact, it is more efficient to send fewer bits, which are the difference from previous value comparing to the sending of the absolute value. The LoWPAN_TCPHC mechanism is based on sending only those fields or parts of fields that do change with respect to the previously sent packet. That is why, the TCP receiver should save the last received segment fields to decompress the next one. For example, the two first bytes of the sequence number field can be elided if they are equal to the value of the previous segment.

If a TCP segment is lost and must be retransmitted, the retransmitted segment should be sent with a mostly compressed header. Then, the TCPHC receiver updates the context.

The TCP compression format is shown in Figure 10 (TCP Header Encoding). The three first bits are used to dispatch between different types of segments. LOWPAN_TCPHC uses two bytes which contain the uncompressed flags of TCP.

Source and destination port numbers are omitted and replaced by a context identifier. The latter should be sent in connection initiation messages and replaces the source and destination port numbers.

The checksum is not compressed and is used by the receiver to check if the decompressed TCP segment is received correctly. To reduce the TCP header length, only these fields which have been changed between two successive segments need to be sent to the receiver. Thus, based on the previously received segment, the receiver reconstructs the original TCP header.

The urgent pointer field is transmitted only when the urgent bit is set. When a receiver sends a duplicated acknowledgment, LOWPAN_TCPHC can compress the TCP header down to six bytes (2 bytes LOWPAN_TCPHC, 1 byte CID, 1 byte acknowledgment number, 2 bytes Checksum).

4.1. TCP ports

TOC

These fields are part of the definition of a stream and they must be constant for all packets in the stream. TCP port numbers can be elided in TCP compressed segments and replaced by a context identifier (CID). The context identifier should be generated by the the first involved LoWPAN node.

4.2. Flags

TOC

Some of the TCP flags are omitted because TCP control messages that set such flags (SYN, PUSH) are sent uncompressed. The uncompressed flags are : PUSH, FIN, Congestion Window Reduced (CWR) and ECN-Echo indication (ECE). These flags are present in the two bytes of the LOWPAN_TCPHC header.

4.3. Sequence and Acknowledgment numbers

TOC

The sequence number specifies the 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 packet by a value between 0 and the MSS (Maximum Segment Size). Thus, the less significant bytes (LSB) are expected to change mush frequently than the most significant bytes (MSB). Thus, it is often enough to send the N less-significant bytes if the 4-N bytes most significant bytes do not change. The decompressor module can deduce the elided bytes from the previously received segments.
For example: The MSS value is 512 bytes and the current sequence number is 0x00f24512. Then, the sequence number of the next segment should be less or equal to 0x00f24712. The compressed segment can sent with only two bytes instead of sending all 4 bytes. The TCP sender can send two bytes 0x4712 and the TCP receives should add the remaining static bytes 0x00f2. Using this method, we reduce to about 50% the length of sequence number or acknowledgment number fields if the MSS value does not exceed 65535 bytes.
The sequence number can be elided if a receiver is just acknowledging data segments and does not send data to the source (i.e., the receiver sends a pure TCP ACK). The same algorithm is used for the compression of the acknowledgment number and only bytes which are changed should be carried in-line. If the TCP sink does not generate data, the four bytes

of the sequence number are omitted in all acknowledgment segments and only compressed acknowledgment fields should be sent.

4.4. Window

TOC

The window field can be omitted if it does not change in time. Moreover, only the window field bits that have been changed should be sent. The decompression deduces the value of this field from the last received full segment.

4.5. Urgent Pointer

TOC

The urgent pointer field is sent in full header format only if the urgent flag is set. Otherwise, this field is elided.

---

5. LOWPAN_TCPHC Packet Format

TOC

5.1. TCP segments types

TOC

Three types of packets are used in a TCP session with header compression:

Regular header TCP segment: A normal, uncompressed header. Does not carry any CID. Figure 7 (Regular header TCP segment) shows the packet format of a regular TCP segment.

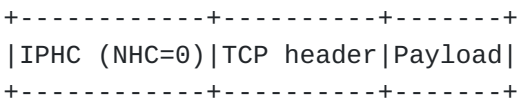


Figure 7: Regular header TCP segment

Full header TCP segment: An uncompressed header that updates or refreshes the context for a packet stream. It carries a CID that

will be used to identify the context. Figure 8 (Full header TCP segment) shows the packet format of a full header TCP segment.

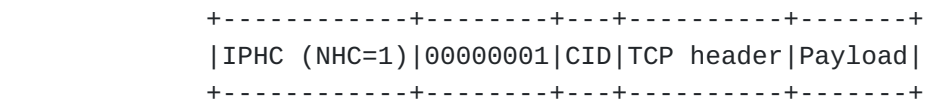


Figure 8: Full header TCP segment

Compressed header TCP segment: Figure 9 (Compressed header TCP segment) shows the header stack of a compressed TCP segment.

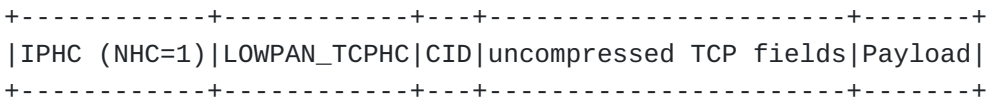


Figure 9: Compressed header TCP segment

5.2. LOWPAN_TCPHC Format

TOC

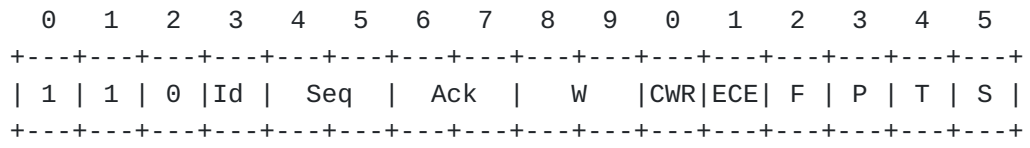


Figure 10: TCP Header Encoding

Id: Context Identifier Size

0: CID is coded in 8 bits.

1: CID is coded in 16 bits.

Seq:

Sequence Number:

00: All 32 bits of Sequence Number are elided.

01: First 8 less-significant bits of Sequence Number are carried in-line. The remaining 24 bits are elided.

10: First 16 less-significant bits of Sequence Number are carried in-line. Last 16 bits of Sequence Number are elided.

11: All 32 bits of Sequence Number are carried in-line.

Ack: Acknowledgment Number:

00: All 32 bits of Acknowledgment Number are elided.

01: First 8 less-significant bits of Acknowledgment Number are carried in-line. The remaining 24 bits are elided.

10: First 16 less-significant bits of Acknowledgment Number are carried in-line. Last 16 bits of Acknowledgment Number are elided.

11: All 32 bits of Acknowledgment Number are carried in-line.

W: Window:

00: The Window field is elided.

01: The less-significant byte of Window field is carried in-line. The most-significant byte is elided.

10: The most-significant byte of Window field is carried in-line. The less-significant byte is elided.

11: Full 16 bits for Window field are carried in-line.

F: FIN flag.

P: PUSH flag.

CWR: Congestion Window Reduced flag.

ECE: ECN-Echo flag.

T: Set if the TCP header contains Timestamp option.

S: Set if the TCP header contains SACK option.

Fields carried in-line (in part or in whole) appear in the same order as they do in the TCP header format [RFC0793] (Postel, J., "Transmission Control Protocol," September 1981.). The TCP Length field must always be elided and it is inferred from lower layers using the 6LoWPAN fragmentation header or the MAC layer header.

5.3. Examples of compressed TCP headers

TOC

In this section, we present some examples of a compressed TCP header using LOWPAN_TCPHC.

5.3.1. Compressed header

TOC

Figure 11 (Compressed header TCP Header Encoding) represents a header of a TCP data segment whose window field has not changed from with respect its antecedent, the two bytes of the lowest bytes of the sequence number that have been changed. The size of this header is seven bytes.

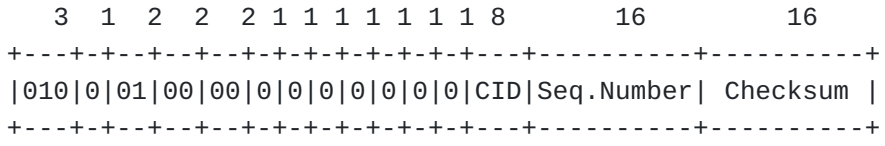


Figure 11: Compressed header TCP Header Encoding

6. TCP Option Compression

TOC

This section defines a compression method for the TCP options most likely to be used in 6LoWPAN. The "end of option" byte and the "no operatioon" byte are elided. Taking into account the caracterisctics of the LLNs, the "window scale" option are not supported because it is especially needed in broadband network and the window size in LLNs are limited to few segments. The "maximum segment size" option is negociated in the first control segments, thus they are not compressed. SACK option are not negociated and are allowed by default. However, the ER can decide to allow or to deny an option sent in the SYN segment.

LOWPAN_TCPHC compresses the mostly used TCP options : SACK and Timestamp. We assume that the SACK and Timestamp are not negotiated and used by default. LOWPAN_TCPHC specifies two bits for SACK and Timestamp TCP options. Figure 12 (TCP Header Option Configuration) shows the structure of a TCP segment including options compressed using LOWPAN_TCPHC. MSS and SACK-Permitted options are sent in a SYN and they are compressed. The Window Scale Option (WSO) is useless in 6LoWPAN because it is more performance to use small windows than large windows. The size of the SACK option is 4 bytes and the size of Timestamp option is variable from 2 to 8 bytes.

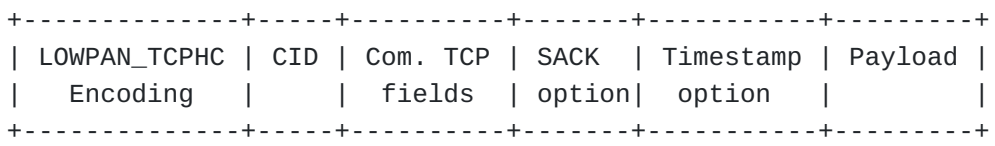


Figure 12: TCP Header Option Configuration

6.1. Selective Acknowledgment option

TOC

The TCP Selective Acknowledgment option (SACK) [RFC2018] (Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, “TCP Selective Acknowledgment Options,” October 1996.) [RFC2883] (Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, “An Extension to the Selective Acknowledgment (SACK) Option for TCP,” July 2000.) should be negotiated in set-up phase, then the option may be used when dropped segments are detected by the receiver. This option is to be used to convey extended acknowledgment information over an established connection. The left edge of the block can be replaced by the offset between the first byte of the segment and the right edge by the length of the block. The Left edge and the right edge will be coded in 16 bits.

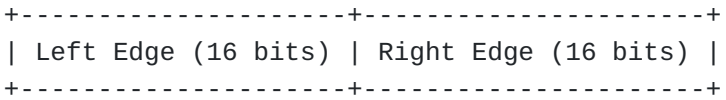


Figure 13: Compressed SACK option

6.2. Timestamp option

TOC

This option carries eight-byte timestamp fields. If timestamp options [RFC1323] (Jacobson, V., Braden, B., and D. Borman, “TCP Extensions for High Performance,” May 1992.) are exchanged in the connection set-up phase, they are expected to appear on all subsequent segments. This overhead added by this option can be reduced: a TCP that does not sent data is not interested to compute the RTTM. And thus, it can replay by sending only Timestamp Echo Reply field (TSecr). However, the Timestamp Value field (TSval) is more important for TCP that send data. LOWPAN_TCPHC sends only bytes that have changed since the last segment to reduce the size of the Timestamp field. LOWPAN_TCPHC defines a bitmap field which specifies the bytes that are elided and the bytes that are carried in-line. Figure 14 (Compressed Timestamp option) shows the structure of the compressed TCP timestamp option fields.

Figure 14: Compressed Timestamp option

7. Acknowledgments

TOC

This work has been funded by the Pole de Recherche Avancée en Communications (PRACom). The authors would like to thank Patrick Maillé and Tiancong Zheng for their useful comments on an early version of this document.

8. References

TOC

[Perkins et al.]	Perkins, S. and M. Multa, “Dependency removal for transport protocol header compression over noisy channels,” International Conference on Communications 1997, June 1997.
[Srivastava et al.]	Srivastava, A., Friday, R., Ritter, M., and W. San Filippo, “A study of TCP performance over wireless data networks,” Vehicular Technology Conference, IEEE VTS 53rd, May 2001.
[Wang04]	Wang, R., “An experimental study of TCP/IP's Van Jacobson header compression behavior in lossy space environment,” Vehicular Technology Conference, IEEE VTC 60th, September 2004.
[Medina05]	Medina, A., Allman, M., and S. Floyd, “Measuring the Evolution of Transport Protocols in the Internet,” ACM SIGCOMM Computer Communications Review 35(2):37-51, April 2005.
[I-D.ietf-6lowpan-hc]	Hui, J. and P. Thubert, “Compression Format for IPv6 Datagrams in 6LoWPAN Networks,” October 2009.
[IEEE 802.15.4]	IEEE Computer Society, “IEEE Std. 802.15.4-2006,” IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks- Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), October 2006.
[RFC0768]	Postel, J., “User Datagram Protocol,” STD 6, RFC 768, August 1980 (TXT).
[RFC0793]	Postel, J., “Transmission Control Protocol,” STD 7, RFC 793, September 1981 (TXT).
[RFC1122]	Braden, R., “Requirements for Internet Hosts - Communication Layers,” STD 3, RFC 1122, October 1989 (TXT).
[RFC1144]	Jacobson, V., “Compressing TCP/IP headers for low-speed serial links,” RFC 1144, February 1990 (TXT, PS, PDF).
[RFC1323]	Jacobson, V., Braden, B., and D. Borman, “TCP Extensions for High Performance,” RFC 1323, May 1992 (TXT).
[RFC2018]	Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, “TCP Selective Acknowledgment Options,” RFC 2018, October 1996 (TXT, HTML, XML).
[RFC2119]	

	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2507]	Degermark, M., Nordgren, B., and S. Pink, “IP Header Compression,” RFC 2507, February 1999 (TXT).
[RFC2883]	Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, “An Extension to the Selective Acknowledgment (SACK) Option for TCP,” RFC 2883, July 2000 (TXT).
[RFC4413]	West, M. and S. McCann, “TCP/IP Field Behavior,” RFC 4413, March 2006 (TXT).
[RFC4919]	Kushalnagar, N., Montenegro, G., and C. Schumacher, “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals,” RFC 4919, August 2007 (TXT).
[RFC4944]	Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” RFC 4944, September 2007 (TXT).
[RFC4995]	Jonsson, L-E., Pelletier, G., and K. Sandlund, “The RObust Header Compression (ROHC) Framework,” RFC 4995, July 2007 (TXT).
[RFC4996]	Pelletier, G., Sandlund, K., Jonsson, L-E., and M. West, “RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP),” RFC 4996, July 2007 (TXT).

Authors' Addresses

TOC

	Ahmed Ayadi
	Telecom Bretagne
	Rue de la Chataigneraie, CS 17607
	35576 Cesson Sevigne cedex
	France
Phone:	+33 2 99 12 70 52
Email:	ahmed.ayadi@telecom-bretagne.eu
	David Ros
	Telecom Bretagne
	Rue de la Chataigneraie, CS 17607
	35576 Cesson Sevigne cedex
	France
Phone:	+33 2 99 12 70 46
Email:	david.ros@telecom-bretagne.eu
	Laurent Toutain
	Telecom Bretagne
	Rue de la Chataigneraie, CS 17607

	35576 Cesson Sevigne cedex
	France
Phone:	+33 2 99 12 70 26
Email:	laurent.toutain@telecom-bretagne.eu