LWIG Working Group
Internet-Draft
Intended status: Informational
Expires: December 31, 2017

C. Gomez
UPC/i2CAT
J. Crowcroft
University of Cambridge
M. Scharf
Nokia

June 29, 2017

TCP over Constrained-Node Networks
draft-gomez-lwig-tcp-constrained-node-networks-03

Abstract

This document provides a profile for the Transmission Control Protocol (TCP) over Constrained-Node Networks (CNNs). The overarching goal is to offer simple measures to allow for lightweight TCP implementation and suitable operation in such environments.

Table of Contents

1. Introduction

The Internet Protocol suite is being used for connecting Constrained-Node Networks (CNNs) to the Internet, enabling the so-called Internet of Things (IoT) [RFC7228]. In order to meet the requirements that stem from CNNs, the IETF has produced a suite of protocols specifically designed for such environments [I-D.ietf-lwig-energy-efficient].

At the application layer, the Constrained Application Protocol (CoAP) was developed over UDP [RFC7252]. However, the integration of some CoAP deployments with existing infrastructure is being challenged by middleboxes such as firewalls, which may limit and even block UDPbased communications. This the main reason why a CoAP over TCP specification is being developed [I-D.tschofenig-core-coap-tcp-tls].

On the other hand, other application layer protocols not specifically designed for CNNs are also being considered for the IoT space. Some examples include HTTP/2 and even HTTP/1.1, both of which run over TCP by default [RFC7540][RFC2616], and the Extensible Messaging and Presence Protocol (XMPP) [RFC 6120]. TCP is also used by non-IETF application-layer protocols in the IoT space such as MQTT and its lightweight variants [MQTTS].

This document provides a profile for TCP over CNNs. The overarching goal is to offer simple measures to allow for lightweight TCP implementation and suitable operation in such environments.

1.1. Conventions used in this document

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]

2. Characteristics of CNNs relevant for TCP

CNNs are defined in [RFC7228] as networks whose characteristics are influenced by being composed of a significant portion of constrained nodes. The latter are characterized by significant limitations on processing, memory, and energy resources, among others [RFC7228]. The first two dimensions pose constraints on the complexity and on the memory footprint of the protocols that constrained nodes can support. The latter requires techniques to save energy, such as radio duty-cycling in wireless devices [I-D.ietf-lwig-energy-efficient], as well as minimization of the number of messages transmitted/received (and their size).

Constrained nodes often use physical/link layer technologies that have been characterized as ’lossy’. Many such technologies are wireless, therefore exhibiting a relatively high bit error rate. However, some wired technologies used in the CNN space are also lossy (e.g. Power Line Communication). Transmission rates of CNN radio or wired interfaces are typically low (e.g. below 1 Mbps).

Some CNNs follow the star topology, whereby one or several hosts are linked to a central device that acts as a router connecting the CNN to the Internet. CNNs may also follow the multihop topology [RFC6606].

3. Scenario

The main scenario for use of TCP over CNNs comprises a constrained device and an unconstrained device that communicate over the Internet using TCP, possibly traversing a middlebox (e.g. a firewall, NAT, etc.). Figure 1 illustrates such scenario. Note that the scenario is asymmetric, as the unconstrained device will typically not suffer the severe constraints of the constrained device. The unconstrained device is expected to be mains-powered, to have high amount of memory and processing power, and to be connected to a resource-rich network.

Assuming that a majority of constrained devices will correspond to sensor nodes, the amount of data traffic sent by constrained devices (e.g. sensor node measurements) is expected to be higher than the amount of data traffic in the opposite direction. Nevertheless, constrained devices may receive requests (to which they may respond), commands (for configuration purposes and for constrained devices including actuators) and relatively infrequent firmware/software updates.

```
                                                       +---------------+
         o     o <--------- TCP communication ------>  |               |
        o     o                                        |               |
           o     o                                     | Unconstrained |
     o        o             +-----------+              |    device     |
         o     o   o  ------ | Middlebox |  -------    |               |
          o   o             +-----------+              |  (e.g. cloud) |
        o    o  o                                      |               |
                                                       +---------------+
    constrained devices
```

Figure 1: TCP communication between a constrained device and an unconstrained device, traversing a middlebox.

4. TCP over CNNs

4.1. TCP connection initiation

In the constrained device to unconstrained device scenario illustrated above, a TCP connection is typically initiated by the constrained device, in order for this device to support possible sleep periods to save energy.

4.2. Maximum Segment Size (MSS)

Some link layer technologies in the CNN space are characterized by a short data unit payload size, e.g. up to a few tens or hundreds of bytes. For example, the maximum frame size in IEEE 802.15.4 is 127 bytes.

6LoWPAN defined an adaptation layer to support IPv6 over IEEE 802.15.4 networks. The adaptation layer includes a fragmentation mechanism, since IPv6 requires the layer below to support an MTU of 1280 bytes [RFC2460], while IEEE 802.15.4 lacked fragmentation mechanisms. 6LoWPAN defines an IEEE 802.15.4 link MTU of 1280 bytes [RFC4944]. Other technologies, such as Bluetooth LE [RFC7668], ITU-T G.9959 [RFC7428] or DECT-ULE [RFC8105], also use 6LoWPAN-based adaptation layers in order to enable IPv6 support. These technologies do support link layer fragmentation. By exploiting this functionality, the adaptation layers that enable IPv6 over such technologies also define an MTU of 1280 bytes.

For devices using technologies with a link MTU of 1280 bytes (e.g. defined by a 6LoWPAN-based adaptation layer), in order to avoid IP layer fragmentation, the TCP MSS must not be set to a value greater than 1220 bytes in CNNs, and it must not be set to a value leading to an IPv6 datagram size exceeding 1280 bytes. (Note: IP version 6 is assumed.)

On the other hand, there exist technologies also used in the CNN space, such as Master Slave / Token Passing (TP) [RFC8163], Narrowband IoT (NB-IoT) [I-D.ietf-lpwan-overview] or IEEE 802.11ah [I-D.delcarpio-6lo-wlanah], that do not suffer the same degree of frame size limitations as the technologies mentioned above. The MTU for MS/TP is recommended to be 1500 bytes [RFC8163], the MTU in NB-IoT is 1600 bytes, and the maximum frame payload size for IEEE 802.11ah is 7991 bytes. Over such technologies, the TCP MSS may be set to a value greater than 1220 bytes, as long as IPv6 datagram size does not exceed the MTU for each technology. One consideration in this regard is that, when a node supports an MTU greater than 1280 bytes, it 'SHOULD' then support Path MTU (PMTU) discovery [RFC1981]. (Note that, as explained in RFC 1981, a minimal IPv6 implementation may 'choose to omit implementation of Path MTU Discovery'). For the sake of lightweight implementation and operation, unless applications require handling large data units (i.e. leading to an IPv6 datagram size greater than 1280 bytes), it may be desirable to limit the MTU to 1280 bytes.

4.3. Window Size

A TCP stack can reduce the implementation complexity by advertising a TCP window size of one MSS, and also transmit at most one MSS of unacknowledged data, at the cost of decreased performance. This size for receive and send window is appropriate for simple message exchanges in the CNN space, reduces implementation complexity and memory requirements, and reduces overhead (see section 4.7).

A TCP window size of one MSS follows the same rationale as the default setting for NSTART in [RFC7252], leading to equivalent operation when CoAP is used over TCP.

For devices that can afford greater TCP window size, it may be useful to allow window sizes of at least five MSSs, in order to allow Fast Retransmit and Fast Recovery [RFC5681].

4.4. RTO estimation

If a TCP sender uses very small window size and cannot use Fast Retransmit/Fast Recovery or SACK, the RTO algorithm has a larger impact on performance than for a more powerful TCP stack. In that case, RTO algorithm tuning may be considered, although careful assessment of possible drawbacks is recommended. A fundamental trade-off exists between responsiveness and correctness of RTOs [I-D.ietf-tcpm-rto-consider]. A more aggressive RTO behavior reduces wait time before retransmissions, but it also increases the probability of incurring spurious timeouts. The latter lead to unnecessary waste of potentially scarce resources in CNNs such as energy and bandwidth.

On a related note, there has been recent activity in the area of defining an adaptive RTO algorithm for CoAP (over UDP). As shown in experimental studies, the RTO estimator for CoAP defined in [I-D.ietf-core-cocoa] (hereinafter, CoCoA RTO) outperforms state-of-art algorithms designed as improvements to RFC 6298 [RFC6298] for TCP, in terms of packet delivery ratio, settling time after a burst of messages, and fairness (the latter is specially relevant in multihop networks connected to the Internet through a single device, such as a 6LoWPAN Border Router (6LBR) configured as a RPL root) [Commag]. In fact, CoCoA RTO has been designed specifically considering the challenges of CNNs, in contrast with the RFC 6298 RTO.

4.5. TCP connection lifetime

[[Note: future revisions will better separate what a TCP stack should support, or not, and how the TCP stack should be used by applications, e.g., whether to close connections or not.]]

4.5.1. Long TCP connection lifetime

In CNNs, in order to minimize message overhead, a TCP connection should be kept open as long as the two TCP endpoints have more data to exchange or it is envisaged that further segment exchanges will take place within an interval of two hours since the last segment has been sent. A greater interval may be used in scenarios where applications exchange data infrequently.

TCP keep-alive messages [RFC1122] may be supported by a server, to check whether a TCP connection is active, in order to release state of inactive connections. This may be useful for servers running on memory-constrained devices.

Since the keep-alive timer may not be set to a value lower than two hours [RFC1122], TCP keep-alive messages are not useful to guarantee that filter state records in middleboxes such as firewalls will not be deleted after an inactivity interval typically in the order of a few minutes [RFC6092]. In scenarios where such middleboxes are present, alternative measures to avoid early deletion of filter state records (which might lead to frequent establishment of new TCP connections between the two involved endpoints) include increasing the initial value for the filter state inactivity timers (if possible), and using application layer heartbeat messages.

4.5.2. Short TCP connection lifetime

A different approach to addressing the problem of traversing middleboxes that perform early filter state record deletion relies on using TCP Fast Open (TFO) [RFC7413]. In this case, instead of trying to maintain a TCP connection for long time, possibly short-lived connections can be opened between two endpoints while incurring low overhead. In fact, TFO allows data to be carried in SYN (and SYN-ACK) packets, and to be consumed immediately by the receceiving endpoint, thus reducing overhead compared with the traditional three-way handshake required to establish a TCP connection.

For security reasons, TFO requires the TCP endpoint that will open the TCP connection (which in CNNs will typically be the constrained device) to request a cookie from the other endpoint. The cookie, with a size of 4 or 16 bytes, is then included in SYN packets of subsequent connections. The cookie needs to be refreshed (and obtained by the client) after a certain amount of time. Nevertheless, TFO is more efficient than frequently opening new TCP connections (by using the traditional three-way handshake) for transmitting new data, as long as the cookie update rate is well below the data new connection rate.

4.6. Explicit congestion notification

Explicit Congestion Notification (ECN) [RFC3168] may be used in CNNs. ECN allows a router to signal in the IP header of a packet that congestion is arising, for example when queue size reaches a certain threshold. If such a packet encapsulates a TCP data packet, an ECN-enabled TCP receiver will echo back the congestion signal to the TCP sender by setting a flag in its next TCP ACK. The sender triggers congestion control measures as if a packet loss had happened. In that case, when the congestion window of a TCP sender has a size of one segment, the TCP sender resets the retransmit timer, and will only be able to send a new packet when the retransmit timer expires [RFC3168]. Effectively, the TCP sender reduces at that moment its sending rate from 1 segment per Round Trip Time (RTT) to 1 segment per default RTO.

ECN can reduce packet losses, since congestion control measures can be applied earlier than after the reception of three duplicate ACKs (if the TCP sender window is large enough) or upon TCP sender RTO expiration [RFC2884]. Therefore, the number of retries decreases, which is particularly beneficial in CNNs, where energy and bandwidth resources are typically limited. Furthermore, latency and jitter are also reduced.

ECN is particularly appropriate in CNNs, since in these environments transactional type interactions are a dominant traffic pattern. As transactional data size decreases, the probability of detecting congestion by the presence of three duplicate ACKs decreases. In contrast, ECN can still activate congestion control measures without requiring three duplicate ACKs.

4.7. TCP options

A TCP implementation needs to support options 0, 1 and 2 [RFC793]. A TCP implementation for a constrained device that uses a single-MSS TCP receive or transmit window size may not benefit from supporting the following TCP options: Window scale [RFC1323], TCP Timestamps [RFC1323], Selective Acknowledgements (SACK) and SACK-Permitted [RFC2018]. Other TCP options should not be used, in keeping with the principle of lightweight operation.

Other TCP options should not be supported by a constrained device, in keeping with the principle of lightweight implementation and operation.

If a device, with less severe memory and processing constraints, can afford advertising a TCP window size of several MSSs, it may support the SACK option to improve performance. SACK allows a data receiver to inform the data sender of non-contiguous data blocks received, thus a sender (having previously sent the SACK-Permitted option) can avoid performing unnecessary retransmissions, saving energy and bandwidth, as well as reducing latency. The receiver supporting SACK will need to manage the reception of possible out-of-order received segments, requiring sufficient buffer space.

SACK adds 8*n+2 bytes to the TCP header, where n denotes the number of data blocks received, up to 4 blocks. For a low number of out-of-order segments, the header overhead penalty of SACK is compensated by avoiding unnecessary retransmissions.

Another potentially relevant TCP option in the context of CNNs is (TFO) [RFC7413]. As described in section 4.5.2, TFO can be used to address the problem of traversing middleboxes that perform early filter state record deletion.

4.8. Delayed Acknowledgments

A device that advertises a single-MSS receive window needs to avoid use of delayed ACKs in order to avoid contributing unnecessary delay (of up to 500 ms) to the RTT [RFC5681].

When traffic over a CNN is expected to be mostly of transactional type, with transaction size typically below one MSS, delayed ACKs are not recommended. For transactional-type traffic between a constrained device and a peer (e.g. backend infrastructure) that uses delayed ACKs, the maximum ACK rate of the peer will be typically of one ACK every 200 ms (or even lower). If in such conditions the peer device is administered by the same entity managing the constrained device, it is recommended to disable delayed ACKs at the peer side.

On the other hand, delayed ACKs allow to reduce the number of ACKs in bulk transfer type of traffic, e.g. for firmware/software updates or for transferring larger data units containing a batch of sensor readings.

4.9. Explicit loss notifications

There has been a significant body of research on solutions capable of explicitly indicating whether a TCP segment loss is due to corruption, in order to avoid activation of congestion control mechanisms [ETEN] [RFC2757]. While such solutions may provide significant improvement, they have not been widely deployed and remain as experimental work. In fact, as of today, the IETF has not standardized any such solution.

5. Security Considerations

If TFO is used, the security considerations of RFC 7413 apply.

There exist TCP options which improve TCP security. Examples include the TCP MD5 signature option [RFC2385] and the TCP Authentication Option (TCP-AO) [RFC5925]. However, both options add overhead and complexity. The TCP MD5 signature option adds 18 bytes to every segment of a connection. TCP-AO typically has a size of 16-20 bytes.

6. Acknowledgments

Carles Gomez has been funded in part by the Spanish Government (Ministerio de Educacion, Cultura y Deporte) through the Jose Castillejo grant CAS15/00336 and by European Regional Development Fund (ERDF) and the Spanish Government through project TEC2016-79988-P, AEI/FEDER, UE. Part of his contribution to this work has been carried out during his stay as a visiting scholar at the Computer Laboratory of the University of Cambridge.

The authors appreciate the feedback received for this document. The following folks provided comments that helped improve the document: Carsten Bormann, Zhen Cao, Wei Genyu, Michael Scharf, Ari Keranen, Abhijan Bhattacharyya, Andres Arcia-Moret, Yoshifumi Nishida, Joe Touch, Fred Baker, Nik Sultana, Kerry Lynn, and Erik Nordmark. Simon Brummer provided details on the RIOT TCP implementation. Xavi Vilajosana provided details on the OpenWSN TCP implementation.

7. Annex. TCP implementations for constrained devices

This section overviews the main features of TCP implementations for constrained devices.

7.1. uIP

uIP is a TCP/IP stack, targetted for 8 and 16-bit microcontrollers. uIP has been deployed with Contiki and the Arduino Ethernet shield.

A code size of ~5 kB (which comprises checksumming, IP, ICMP and TCP) has been reported for uIP [Dunk].

uIP provides a global buffer for incoming packets, of single-packet size. A buffer for outgoing data is not provided. In case of a retransmission, an application must be able to reproduce the same packet that had been transmitted.

The MSS is announced via the MSS option on connection establishment and the receive window size (of one MSS) is not modified during a connection. Stop-and-wait operation is used for sending data. Among other optimizations, this allows to avoid sliding window operations, which use 32-bit arithmetic extensively and are expensive on 8-bit CPUs.

7.2. lwIP

lwIP is a TCP/IP stack, targetted for 8- and 16-bit microcontrollers. lwIP has a total code size of ~14 kB to ~22 kB (which comprises memory management, checksumming, network interfaces, IP, ICMP and TCP), and a TCP code size of ~9 kB to ~14 kB [Dunk].

In contrast with uIP, lwIP decouples applications from the network stack. lwIP supports a TCP transmission window greater than a single segment, as well as buffering of incoming and outcoming data. Other implemented mechanisms comprise slow start, congestion avoidance, fast retransmit and fast recovery. SACK and Window Scale have been recently added to lwIP.

7.3. RIOT

The RIOT TCP implementation (called GNRC TCP) has been designed for Class 1 devices [RFC 7228]. The main target platforms are 8- and 16-bit microcontrollers. GNRC TCP offers a similar function set as uIP, but it provides and maintains an independent receive buffer for each connection. In contrast to uIP, retransmission is also handled by GNRC TCP. GNRC TCP uses a single-MSS window size, which simplifies the implementation. The application programmer does not need to know anything about the TCP internals, therefore GNRC TCP can be seen as a user-friendly uIP TCP implementation.

The MSS is set on connections establishment and cannot be changed during connection lifetime. GNRC TCP allows multiple connections in parallel, but each TCB must be allocated somewhere in the system. By default there is only enough memory allocated for a single TCP connection, but it can be increased at compile time if the user needs multiple parallel connections.

7.4. OpenWSN

The TCP implementation in OpenWSN is mostly equivalent to the uIP TCP implementation. OpenWSN TCP implementation only supports the minimum state machine functionality required. For example, it does not perform retransmissions.

7.5. TinyOS

TBD

7.6. Summary

		uIP	lwIP orig	lwIP 2.0	RIOT	OpenWSN	TinyOS
Memory	Data size	*	*	*	*	*	*
	Code size (kB)	< 5	~9 to ~14	*	*	*	*
TCP features	Window size(MSS)	1	Multiple	Multiple	1	1	*
	Slow start	No	Yes	Yes	No	No	*
	Fast rec/retx	No	Yes	Yes	No	No	*
	Keep-alive	No	*	*	No	No	*
	TFO	No	No	*	No	No	*
	ECN	No	No	*	No	No	*
	Window Scale	No	No	Yes	No	No	*
	TCP timestamps	No	No	Yes	No	No	*
	SACK	No	No	Yes	No	No	*

	Delayed ACKs	No	Yes	Yes	No	No	*

Figure 2: Summary of TCP features for differrent lightweight TCP implementations.

8. References

8.1. Normative References

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, DOI 10.17487/RFC1323, May 1992, <http://www.rfc-editor.org/info/rfc1323>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, DOI 10.17487/RFC2018, October 1996, <http://www.rfc-editor.org/info/rfc2018>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, DOI 10.17487/RFC2385, August 1998, <http://www.rfc-editor.org/info/rfc2385>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <http://www.rfc-editor.org/info/rfc2616>.

[RFC2757] Montenegro, G., Dawkins, S., Kojo, M., Magret, V., and N. Vaidya, "Long Thin Networks", RFC 2757, DOI 10.17487/RFC2757, January 2000, <http://www.rfc-editor.org/info/rfc2757>.

[RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, DOI 10.17487/RFC2884, July 2000, <http://www.rfc-editor.org/info/rfc2884>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <http://www.rfc-editor.org/info/rfc4944>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <http://www.rfc-editor.org/info/rfc6092>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP’s Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <http://www.rfc-editor.org/info/rfc6298>.

[RFC6606] Kim, E., Kaspar, D., Gomez, C., and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing", RFC 6606, DOI 10.17487/RFC6606, May 2012, <http://www.rfc-editor.org/info/rfc6606>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

[RFC7428] Brandt, A. and J. Buron, "Transmission of IPv6 Packets over ITU-T G.9959 Networks", RFC 7428, DOI 10.17487/RFC7428, February 2015, <http://www.rfc-editor.org/info/rfc7428>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC7668] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "IPv6 over BLUETOOTH(R) Low Energy", RFC 7668, DOI 10.17487/RFC7668, October 2015, <http://www.rfc-editor.org/info/rfc7668>.

[RFC8105] Mariager, P., Petersen, J., Ed., Shelby, Z., Van de Logt, M., and D. Barthel, "Transmission of IPv6 Packets over Digital Enhanced Cordless Telecommunications (DECT) Ultra Low Energy (ULE)", RFC 8105, DOI 10.17487/RFC8105, May 2017, <http://www.rfc-editor.org/info/rfc8105>.

[RFC8163] Lynn, K., Ed., Martocci, J., Neilson, C., and S. Donaldson, "Transmission of IPv6 over Master-Slave/Token-Passing (MS/TP) Networks", RFC 8163, DOI 10.17487/RFC8163, May 2017, <http://www.rfc-editor.org/info/rfc8163>.

8.2. Informative References

[Commag] A. Betzler, C. Gomez, I. Demirkol, J. Paradells, "CoAP Congestion Control for the Internet of Things", IEEE Communications Magazine, June 2016.

[Dunk] A. Dunkels, "Full TCP/IP for 8-Bit Architectures", 2003.

[ETEN] R. Krishnan et al, "Explicit transport error notification (ETEN) for error-prone wireless and satellite networks", Computer Networks 2004.

[I-D.delcarpio-6lo-wlanah]
Vega, L., Robles, I., and R. Morabito, "IPv6 over 802.11ah", draft-delcarpio-6lo-wlanah-01 (work in progress), October 2015.

[I-D.ietf-core-cocoa]
Bormann, C., Betzler, A., Gomez, C., and I. Demirkol, "CoAP Simple Congestion Control/Advanced", draft-ietf-core-cocoa-01 (work in progress), March 2017.

[I-D.ietf-lpwan-overview]
Farrell, S., "LPWAN Overview", draft-ietf-lpwan-overview-04 (work in progress), June 2017.

[I-D.ietf-lwig-energy-efficient]
Gomez, C., Kovatsch, M., Tian, H., and Z. Cao, "Energy-Efficient Features of Internet of Things Protocols", draft-ietf-lwig-energy-efficient-07 (work in progress), March 2017.

[I-D.ietf-tcpm-rto-consider]
Allman, M., "Retransmission Timeout Requirements", draft-ietf-tcpm-rto-consider-05 (work in progress), March 2017.

[I-D.tschofenig-core-coap-tcp-tls]
Bormann, C., Lemay, S., Technologies, Z., and H. Tschofenig, "A TCP and TLS Transport for the Constrained Application Protocol (CoAP)", draft-tschofenig-core-coap-tcp-tls-05 (work in progress), November 2015.

[MQTTS] U. Hunkeler, H.-L. Truong, A. Stanford-Clark, "MQTT-S: A Publish/Subscribe Protocol For Wireless Sensor Networks", 2008.

Authors’ Addresses

Carles Gomez
UPC/i2CAT
C/Esteve Terradas, 7
Castelldefels 08860
Spain

Email: carlesgo@entel.upc.edu

Jon Crowcroft
University of Cambridge
JJ Thomson Avenue
Cambridge, CB3 0FD
United Kingdom

Email: jon.crowcroft@cl.cam.ac.uk

Michael Scharf
Nokia
Lorenzstrasse 10
Stuttgart, 70435
Germany

Email: michael.scharf@nokia.com