

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.

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 December 31, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Conventions used in this document	3
2. Characteristics of CNNs relevant for TCP	3
3. Scenario	4
4. TCP over CNNs	4
4.1. TCP connection initiation	4
4.2. Maximum Segment Size (MSS)	5
4.3. Window Size	6
4.4. RTO estimation	6
4.5. TCP connection lifetime	7
4.5.1. Long TCP connection lifetime	7
4.5.2. Short TCP connection lifetime	7
4.6. Explicit congestion notification	8
4.7. TCP options	8
4.8. Delayed Acknowledgments	9
4.9. Explicit loss notifications	10
5. Security Considerations	10
6. Acknowledgments	10
7. Annex. TCP implementations for constrained devices	10
7.1. uIP	10
7.2. lwIP	11
7.3. RIOT	11
7.4. OpenWSN	12
7.5. TinyOS	12
7.6. Summary	12
8. References	13
8.1. Normative References	13
8.2. Informative References	15
Authors' Addresses	16

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 UDP-

based 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 [MQTT5].

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.

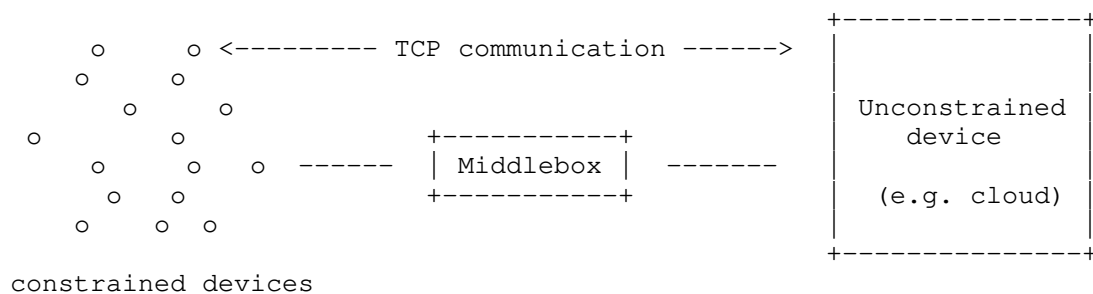


Figure 1: TCP communication between a constrained device and an unconstrained device, traversing a middlebox.

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 receiving 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 outgoing 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	Tiny
OS	Data size	*	*	*	*	*	*
	Memory						
	Code size (kB)	< 5	~9 to ~14	*	*	*	*
	Window size (MSS)	1	Multiple	Multiple	1	1	*
T	Slow start	No	Yes	Yes	No	No	*
	Fast rec/retx	No	Yes	Yes	No	No	*
P	Keep-alive	No	*	*	No	No	*
	TFO	No	No	*	No	No	*
e	ECN	No	No	*	No	No	*
	Window Scale	No	No	Yes	No	No	*
s	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 different 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

LWIG
Internet-Draft
Intended status: Informational
Expires: July 21, 2017

R. Jadhav, Ed.
R. Sahoo
Huawei Tech
S. Duquennoy
Inria
J. Eriksson
Yanzi Networks
January 17, 2017

Neighbor Management Policy for 6LoWPAN
draft-jadhav-lwig-nbr-mgmt-policy-00

Abstract

This document describes the problems associated with neighbor cache management in constrained multihop networks and a sample neighbor management policy to deal with it.

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 July 21, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language and Terminology	4
2. Neighbor Management	4
2.1. Significance of Neighbor management policy	4
2.2. Trivial neighbor management policies	5
2.3. Lifecycle of a NCE	6
2.3.1. NCE Insertion	6
2.3.2. NCE Deletion	9
2.3.3. NCE Eviction	10
2.3.3.1. Eviction for directly connected routing entries .	10
2.3.4. NCE Reinforcement	11
2.4. Requirements of a good neighbor management policy	11
2.5. Approaches to neighbor management policy	11
2.5.1. Reactive Approach	12
2.5.2. Proactive Approach	12
3. Reservation based Neighbor Management Policy	13
3.1. Limitations of such a policy	14
4. Acknowledgements	15
5. IANA Considerations	15
6. Security Considerations	15
7. References	15
7.1. Normative References	15
7.2. Informative References	15
Appendix A. Additional Stuff	16
Authors' Addresses	16

1. Introduction

In a wireless multihop network, the node densities (maximum number of devices connected on a single hop) may vary significantly depending upon deployments/scenarios. While there is some policy control possible with regards to the network size in terms of maximum number of devices connected, it is especially difficult to set a figure on what will be the maximum node density given a deployment. For e.g. A network can put an upper limit on max 1000 devices but it is impossible to state what the node density will be in this 1000 node network.

A neighbor cache is used for populating neighboring one-hop connected nodes information such as MAC address, link local IP address and other reachability state information. Node density has direct implications on the neighbor cache and in constrained network scenario the size of the neighbor cache will be limited. Thus there

are chances that a node may not be able to fit all the neighboring nodes in its cache in which case it has to prioritize entries and thus needs a neighbor management policy.

This draft presents problems related to neighbor management policies by considering a security-enabled multi-hop 6Lo network. This document considers RPL [RFC6550] as a routing protocol and PANA (EAP-PANA) [RFC5191] as a network access protocol. For RPL, both the storing and non-storing mode of operations are considered. We also provide a sample neighbor management policy which can be used in such networks and its limitations. The aim of such a policy is to retain set of neighbor cache entries with high quality links such that routing adjacencies are stabilized.

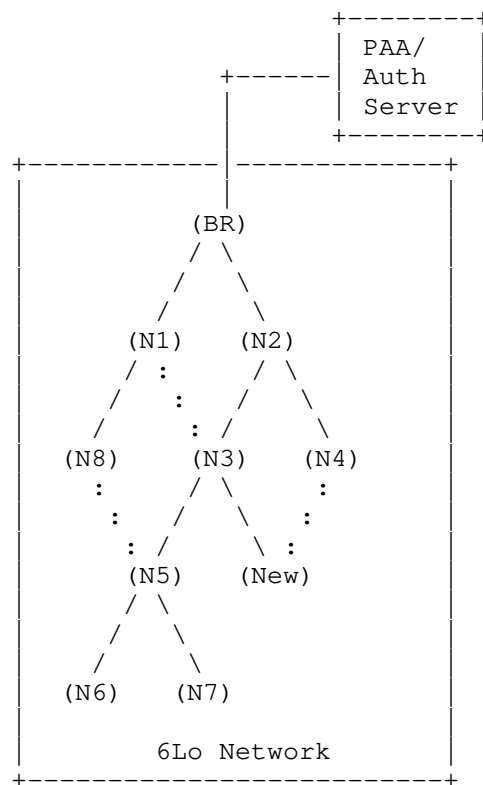


Figure 1: Sample Topology

1.1. Requirements Language and Terminology

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 RFC 2119 [RFC2119].

PaC (PANA Client): New joining node which is yet to be authenticated.

PRE (PANA Relay Element): An already authenticated and network joined node which is willing to act as a relay element for PaCs to complete their authentication procedure on multi-hop networks. [RFC6345] describes the details of PRE.

PAA (PANA Auth Agent): Auth server which hosts the credentials database. PaC will handshake with PAA to complete authentication procedure.

Routing Child: A downstream node who is part of the routing table of the parent. For e.g. in the sample topology above N5 is the directly connected routing child for N3. N6 and N7 are also part of N3 routing table, they are routing child nodes but not directly connected. For N6 and N7 the document might alternatively use a term grand-child.

Routing Parent: In Figure 1, N1 and N2 are possible routing parents for N3.

Neighbor Cache Entry (NCE): A neighbor entry managed on behalf of directly connected peer.

This document also uses terminology described in [RFC6550] and [RFC6775].

2. Neighbor Management

2.1. Significance of Neighbor management policy

Multihop mesh networks present unique challenges to neighbor management especially with resource constrained nodes. In cases where the node density is higher than the neighbor cache size, the entries have to be prioritized. [Woo_et_al] and [Dawans_et_al] talk about prioritization of neighbor entries by using link quality estimation techniques. But prioritization alone may not necessarily be optimal in all cases. The reason or function why neighbor entry was added also needs to be taken in consideration. For example, evicting a routing direct child might have a ripple effect in turn impacting all the sub-children as well.

In case of key management protocols deployed above MAC layer in multihop network, the neighbor management kicks in early even before the routing adjacencies are established. Since a new joining node needs to discover/attach to a relay element for completing its authentication procedure, the neighbor cache entries have to be appropriately populated both on a PaC and on the PRE. If a neighbor entry whose authentication is in progress is evicted, it will negatively impact the authentication procedure.

Another important consideration is that with increased node density, the prioritization based on link estimation parameters might not help since there might be more well connected peers. In dense deployments the number of directly attached neighbors with good quality links might still be higher than the max entries in neighbor cache size.

2.2. Trivial neighbor management policies

This section investigates policies which are used by most of the current operating systems for constrained nodes. While such policies are trivial to implement they may not be able to deal with the constrained network scenario. Note that such policies can still be used if it is known apriori that the neighbor cache can hold entries for maximum node density.

- a. First Come First Serve (FCFS) policy
- b. Least Recently Used (LRU) policy

The primary distinction between these policies is how it treats a new entry when the neighbor cache is full. In case of FCFS policy, the new entry is simply rejected while with LRU, the new entry replaces the least recently used entry.

RPL works by initiating a downstream multicast DIO to establish upstream network path. Subsequently DAO messages might be sent by the nodes to establish downstream paths to the nodes. Thus the network is flooded with multicast DIO messages initially and similarly there are chances that the same node is ended up been selected as a preferred parent by most of the child nodes and thus receives a DAO message from all these child nodes. Note that once a node establishes a parent entry or a routing entry on behalf of a directly connected node then it has to also provision a neighbor cache entry for it for subsequent unicast traffic.

In case of FCFS policy, a node might end up hosting all the neighbor entries based on DIO or DAO messages. Once the cache is full all the subsequent attempts to add an NCE will fail.

In case of LRU policy, a node might end up churning lot of neighbor entries because once the cache gets full and there is a request for new entry, it would result in evicting the least recently used (but active) entry. If at later point of time, there is a traffic for the evicted entry then the old entry has to be reinstated using IPv6 NDP procedure. This would mean reinstating the entry by evicting another least recently used entry. If the node density is very high, then this churn would be substantially high to extent that it would disrupt any routing adjacencies to be established in the network in a stable way.

2.3. Lifecycle of a NCE

2.3.1. NCE Insertion

IPv6 NDP [RFC6775] defines signaling involved in resolving the IPv6 addresses to its corresponding MAC addresses which gets populated in the neighbor cache. In case of constrained network, it is desired that such control traffic is minimized and thus the neighbor cache entries are populated as part of existing messaging. One example would be when the node receives a DAO message from its immediate child node, it not only makes an addition to the routing table but also creates a neighbor cache entry for the node. Thus it eliminates need for additional IPv6 NDP NS/NA messaging involved to resolve MAC address. Similar heuristic is used to add neighbor entries in other cases as well. Section 10.3.2 of [RFC6775] describes update and addition of such NCEs based on routing information packets.

Following are the possible signaling scenarios in which case a neighbor entry may get added.

Node Joining procedure: A new joinee node discovers a relay element to initiate its auth procedure. At the end of the discovery phase the new joinee node would have known the link local IP address of the relay element. The joinee node will send an unsecured-NS to the relay element to solicit its NA. The PRE may send a NA with the suitable status code as defined in section 6.5.3 of [RFC6775].

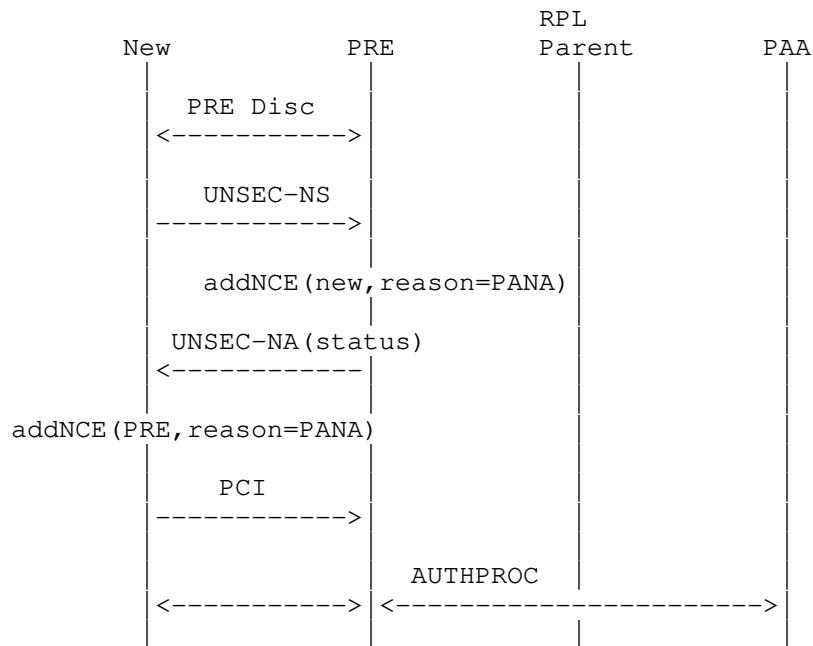


Figure 2: NCE creation between PaC and PRE during relay discovery process

Relay element does not hold any state information on behalf of the new joinee node except for its neighbor cache entry. Thus in the Figure 1 the new joinee node may select node N3 as its PRE, in which case N3 has to add a neighbor entry on behalf of the new joinee node.

Post authentication the node enters into network discovery phase. The node selects one or more of its neighboring peer as its preferred parent based on the DIO received from these peers. Note that the node's selected relay element and its preferred parent may not be same. The preferred parent serves as a default router node to which all its upstream traffic is directed. Thus an NCE on behalf of preferred parent needs to be added. In Figure 1 node N5 selects N3 as its preferred parent. N5 needs to add neighbor entry on behalf of N3 which is its directly connected RPL preferred parent.

In case of RPL storing MOP (mode of operation), the node may send a DAO message containing its reachability information to its preferred parent. The parent node in turn may pass this information upstream to its parent by generating a DAO retaining the child node's reachability information, establishing a downstream routing path towards the node who originated the DAO. The preferred parent has to maintain a neighbor entry on behalf of the directly connected child

node. For example, in the Figure 1, node N3 needs to maintain a neighbor entry on behalf of N5 which is its directly connected child node. Nodes N6 and N7 are grand-child nodes for node N3 for whom no neighbor entry is required.

As mentioned in Section 10.3.2 of [RFC6775], the NCEs on parent and child can be added directly as a result of RPL DIO/DAO signalling without any explicit NS/NA messaging.

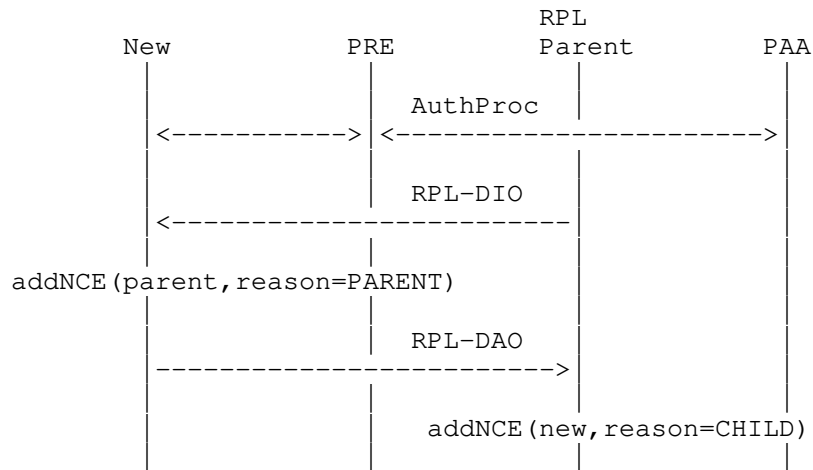


Figure 3: NCE creation call Flow for RPL storing MOP

In case of non-storing MOP, the parent node needs to know the global IPv6 address of the immediate child nodes. This is needed since the source routing header carries the global addresses and thus the NCE of the child node should contain the global address. Secondly, the RPL DAO is addressed directly to the root node in case of non-storing mode. Thus RPL messaging cannot be used for creating NCE entries on parent and child, unlike storing MOP. The child node may send a secure unicast NS with ARO option containing its global address to be registered on the parent node. The child node can still use RPL DIO to create an NCE on behalf of the parent node.

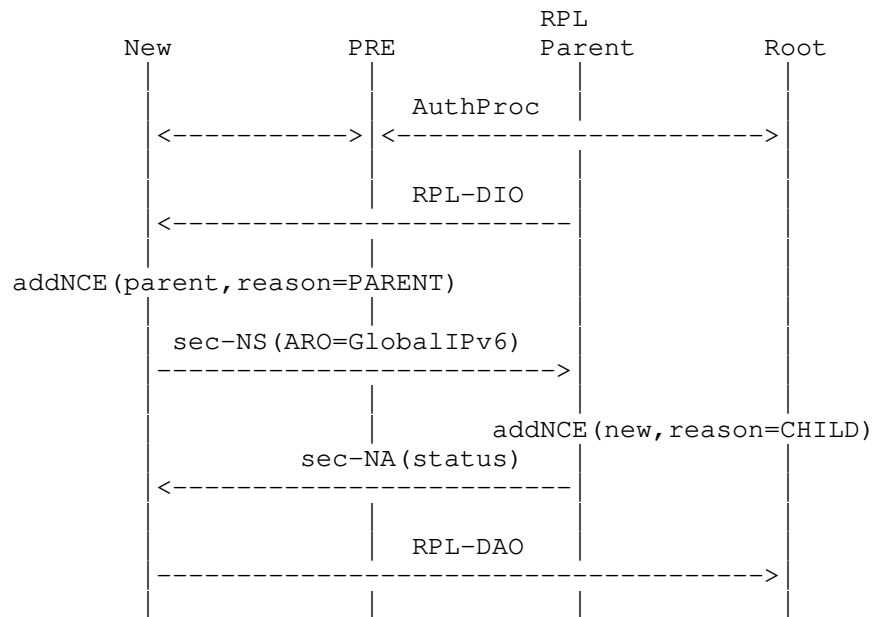


Figure 4: NCE creation call Flow for non-storing MOP

This document expects the neighbor management policy to remember the reason why the neighbor entry is inserted. Secondly, the router may remember whether the NS received was secured or unsecured and accordingly use it to prioritize eviction entries. As described in the next sections, this reason will help the policy to prioritize the entries in case an eviction is required.

2.3.2. NCE Deletion

It is imperative that an unwanted neighbor entry be removed as soon as possible. This section talks about different cases in which neighbor entry can be deleted.

Route Invalidation: In case of storing MOP, when the child node decides to switch its preferred parent, the RPL specifications allows the node to send a no-path DAO message to invalidate the route along the previous path(s). A directly connected parent node can use this message to clear the NCE. While the entry can be immediately cleared, usually the implementations choose to wait a small amount of time before clearing the entry. This is to avoid any impact on the in-transit traffic. Thus this also establishes the importance of route invalidation to achieve optimized neighbor cache utilization.

In case of non-storing mode, the no-path DAO cannot be not employed since the previous parent does not having any routing information to be invalidated. But the previous parent may still contain the NCE on behalf of the child node. This document recommends use of [RFC6775] section 6.5.3. which allows sending a zero lifetime ARO option in NS for deregistering the corresponding neighbor entry.

[RFC6775], ND optimizations for 6LoWPANs, section 5.5.3. talks about deleting the entries in case the NUD (neighbor unreachability detection) fails either due to no response to NS messages or due to failure response. NCEs in such cases should be deleted. An example where NUD NS would fail because of no response is the case where the child node switches its parent due to link unavailability. The parent in such a case would not receive the no-path DAO message or any other traffic from the child node. Thus on NCE lifetime expiry, the parent node would send NS which would fail with no response, thus triggering entry deletion.

2.3.3. NCE Eviction

The eviction rules have a major impact on the neighbor management policy. Eviction rules are used when the policy has to forcibly remove an active neighbor entry from the cache to make space for the new (hopefully higher priority) entry. The eviction policy may take into account several considerations such as the reason why the entry was made, is the entry in active use currently, how good (for e.g., based on link estimation) the entry currently is.

2.3.3.1. Eviction for directly connected routing entries

This section talks about implications of an eviction in which a parent node decides of evicting a directly connected routing child NCE. In the sample topology Figure 1, lets assume N3 needs to evict N5 from its neighbor cache. In case of RPL's storing MOP, eviction of directly connected routing child NCE also has impact on all the sub-children. Thus not only will it result in impacting N5 but also nodes N6 and N7. It is important to note that such an eviction has less impact on RPL's non-storing MOP i.e. in case of non-storing mode N5 might end up selecting alternate parent N8 and does not result in any additional control overhead for node N6 and N7.

Thus RPL's non-storing MOP provides additional eviction flexibility for a neighbor management policy in terms evicting directly connected child entries.

2.3.4. NCE Reinforcement

It is expected that the latest reachability state and metric information be maintained in context to the NCE. With wireless networks, the neighbor cache entries prioritization may change over a period of time especially the link quality estimation parameters or the routing metrics. Reinforcement refers to updating the parameters in context to the NCEs which helps in prioritizing the entries when it comes to handling eviction. In wireless networks, on reception of incoming packet, the receiver node's physical and MAC layer may derive certain signal reception parameters (such as RSSI, LQI) which can be considered for reinforcement purpose if the corresponding transmitter/source entry in neighbor cache is found. It should be noted that the signal quality parameters may have high variance in 6Lo networks and thus statistical techniques (such as weighted averaging) are usually employed for deciding about a link quality over a period of time. Reinforcement can be achieved using one or more of the following techniques:

Passive Monitoring: Reinforcing the quality parameters using packets received from the source. TrickleDIO, periodic beacons, application traffic etc can be used for such monitoring.

Active Probing: A node may select subset of entries for active probing wherein it sends a message to the neighbor entry's target and can expect a response message back. An example of such probing is [CONTIKI] where unicast DIS is sent to solicit a unicast DIO without impacting the trickle timers. Though it adds a control overhead on the link, periodic probing can help to ascertain connectivity in the absence of any other traffic from the neighboring node.

2.4. Requirements of a good neighbor management policy

Route Stability: Stable NCEs will result in stable routing adjacencies. Thus it is important to avoid unnecessary NCE churn for routing path stability.

Control overhead: A neighbor management policy may have to use signalling messages for policy handling (such as rejection of NCE). It is required that such overhead be kept as low as possible.

2.5. Approaches to neighbor management policy

Neighbor management policy depends upon the neighbor cache space availability and the same can be advertised proactively or can be handled reactively.

2.5.1. Reactive Approach

In this approach, the nodes select their RPL parent or the relay element purely based on link metrics and subsequently when they try to allocate their NCE in the target node, it may fail due to unavailability of the cache space. The failure can be communicated depending upon the signaling involved:

NS failure: Section 6.5.3 of [RFC6775] defines a procedure for NS failure handling in case the router's neighbor cache is full. It results in a unicast NA with ARO status field set to two.

DAO NACK: Section 9.3 of RPL [RFC6550] specifies on how can the parent node react to DAOs from child. In case the parent could not make a NCE on behalf of the child node, a negative ACK with status (between 127-255) should be sent to the child node. The natural reaction of the child node would be to switch to an alternate parent.

PANA Failure: PaC's auth session starts with a PaC discovering a PRE. The discovery procedure is not standardized and can be based upon various factors including signal strength of discovery messages from PRE. Post discovery, the PaC needs to send an unsecured unicast NS message with an ARO containing its link-local IPv6 address. NS helps to determine whether the PRE can allocate an NCE for the PaC. PRE accordingly sends a NA response with appropriate status field.

2.5.2. Proactive Approach

Neighbor cache availability could be proactively advertised by the parent nodes in the DIO messages and in the PRE discovery messages. A child RPL node may additionally use this information from DIO as part of parent selection process. In case of new joinee node, the node may use PRE discovery messages with space availability information to select an appropriate PRE. Proactive signaling of neighbor cache space availability will help the nodes to select the parent node or relay node such that the failure signaling due to cache full event can be reduced.

Currently there is no standard way of signaling such neighbor cache space availability information. RPL's DIO messages carry metric information and can be augmented with neighbor cache space as an additional metric. In case of PRE discovery however there is no standard way of defining this information since the PRE discovery procedure itself is not standardized.

In a wireless or shared bus network, a multicast DIO metric advertisement may reach several child nodes eventually everyone responding by selecting the same parent node causing neighbor cache to be exhausted. Thus the failure handling approaches defined in the Reactive Approach section applies here as well. But importantly the failure signaling will be significantly reduced because of proactive advertisement.

3. Reservation based Neighbor Management Policy

This section defines a sample neighbor management policy, with the primary objective to reduce NCE churn and to ensure stability of routing adjacencies. The scheme uses a reservation based policy to reserve NCEs for:

NCE Entry for	MAX count	Reason
Routing Parent	MAX_ROUTING_PARENT_NCE_NUM	PARENT
Routing child	MAX_ROUTING_CHILD_NCE_NUM	CHILD
Others such as pre-auth sessions	MAX_OTHER_NCE_NUM	OTHER

Table 1: Neighbor Cache Entry reservation

Note that reservation policy depends upon identification of the reason behind making an NCE . In case of pre-auth sessions, the corresponding NCE is created based on the unsecured NS/NA. In case of storing MOP, CHILD_ENT NCEs are created either based on DAO (as shown in Figure 3) or based on secured NS/NA messaging (as shown in Figure 4). In case of non-storing MOP, a secured NS/NA messaging as shown in Figure 4 needs to be used.

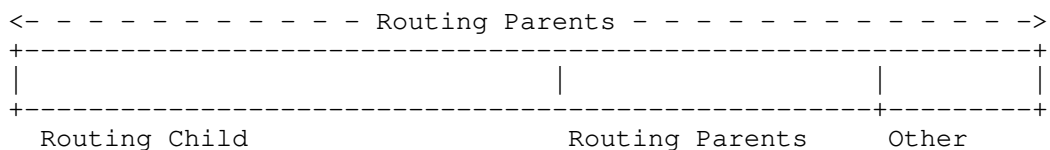


Figure 5: Reservation of NCEs in neighbor table

As shown in the figure, the neighbor cache is partitioned into different entry types. The routing parents can possibly occupy any entry type if found vacant since in case an eviction is sought the non-preferred routing parent could be evicted without much impact on

the functioning or on the control traffic. The eviction could be done based on reasons specified in Section 2.3.3.

Routing Child entries are made in context to directly connected peers and these entries are not deleted unless they are unreachably or there is any reason for the parent node to believe that it is no longer the preferred parent for the child node. Deletion may happen based on reasons mentioned in Section 2.3.2.

Other entries (OTHER) may be made in response to temporary requirement of making an NCE. One such case is the pre authentication phase where in the relay node makes an entry of the PaC temporarily till the time the authentication phase is completed. The NCE made thus is garbage collected at the end of the lifetime. Also an implementation may choose to keep a lower lifetime for such NCEs depending upon the time taken to complete the authentication process.

3.1. Limitations of such a policy

The reservation based policy mentioned in this section may result in sub-optimal path selection due to lack of NCE resource on the parent nodes. Also the restriction of maximum pre-auth sessions in the form of MAX_OTHER_NCE_NUM limits the maximum relay sessions that can be supported on the relay node.

The reservation policy allows the parent node to reject the child node's DAO or NS. But the child node cannot remember this rejection and may reattempt the same parent after some time depending upon triggers such as reception of DIO from the same parent who rejected it previously. One of the only way to stop the child node from reattempting such parent selection would be to also include a proactive approach wherein the parent node signals its resource availability in the DIO message as mentioned in Section 2.5.2. Such a scheme of signalling parent node's resource availability is currently not standardized.

RPL's storing MOP imposes additional restrictions. One such case is where a child node may have a given parent node as its only parent and that parent node's NCE are all used up. In such a case, the child node would keep on retrying and failing to send a DAO through the parent node. Ideally the parent node could have evicted a least used child node or a child node who has an alternate parent available. Evicting such a child node is a complex process and may increase the control overhead as described in Section 2.3.3.1. Thus the reservation based policy requires that the minimum node density is sufficiently high so that every child finds a parent node in its vicinity with enough resources.

4. Acknowledgements

This template was derived from an initial version written by Pekka Savola and contributed by him to the xml2rfc project.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

Add DoS attacks possibility on NBR table on PRE and what are the mechanisms already defined by standards (such as use of Enforcement Point)

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

7. References

7.1. Normative References

[CONTIKI] Thingsquare, "Contiki: The Open Source OS for IoT", 2012, <<http://www.contiki-os.org>>.

[Dawans_et_al]

Dawans, S., Duquennoy, S., and O. Bonaventure, "On Link Estimation in Dense RPL Deployments", 2012.

[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>>.

[Woo_et_al]

Woo, A., Tong, T., and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks", 2003.

7.2. Informative References

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<http://www.rfc-editor.org/info/rfc3552>>.

- [RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschafenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191, May 2008, <<http://www.rfc-editor.org/info/rfc5191>>.
- [RFC6345] Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., Ed., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA) Relay Element", RFC 6345, DOI 10.17487/RFC6345, August 2011, <<http://www.rfc-editor.org/info/rfc6345>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<http://www.rfc-editor.org/info/rfc6550>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<http://www.rfc-editor.org/info/rfc6775>>.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Rahul Arvind Jadhav (editor)
Huawei Tech
Kundalahalli Village, Whitefield,
Bangalore, Karnataka 560037
India

Phone: +91-080-49160700
Email: rahul.ietf@gmail.com

Rabi Narayan Sahoo
Huawei Tech
Kundalahalli Village, Whitefield,
Bangalore, Karnataka 560037
India

Phone: +91-080-49160700
Email: rabinarayans@huawei.com

Simon Duquennoy
Inria
40 Avenue Halley
Building A
Villeneuve d'Ascq
France

Phone: +33 768227731
Email: simon.duquennoy@inria.fr

Joakim Eriksson
Yanzi Networks

Email: joakime@sics.se

Light-Weight Implementation Guidance (lwig)
Internet-Draft
Intended status: Informational
Expires: April 24, 2019

D. Migault
Ericsson
T. Guggemos
LMU Munich
October 21, 2018

Minimal ESP
draft-mglt-lwig-minimal-esp-07

Abstract

This document describes a minimal implementation of the IP Encapsulation Security Payload (ESP) defined in RFC 4303. Its purpose is to enable implementation of ESP with a minimal set of options to remain compatible with ESP as described in RFC 4303. A minimal version of ESP is not intended to become a replacement of the RFC 4303 ESP, but instead to enable a limited implementation to interoperate with implementations of RFC 4303 ESP.

This document describes what is required from RFC 4303 ESP as well as various ways to optimize compliance with RFC 4303 ESP.

This document does not update or modify RFC 4303, but provides a compact description of how to implement the minimal version of the protocol. If this document and RFC 4303 conflicts then RFC 4303 is the authoritative description.

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 <https://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 24, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Requirements notation

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. Introduction

ESP [RFC4303] is part of the IPsec suite protocol [RFC4301]. IPsec is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity) and limited traffic flow confidentiality.

Figure 1 describes an ESP Packet. Currently ESP is implemented in the kernel of major multi purpose Operating Systems (OS). The ESP and IPsec suite is usually implemented in a complete way to fit multiple purpose usage of these OS. However, completeness of the IPsec suite as well as multi purpose scope of these OS is often performed at the expense of resources, or a lack of performance. As a result, constraint devices are likely to have their own implementation of ESP optimized and adapted to their specificities. With the adoption of IPsec by IoT devices with minimal IKEv2 [RFC7815] and ESP Header Compression (EHC) with [I-D.mglt-ipsecme-diet-esp] or [I-D.mglt-ipsecme-ikev2-diet-esp-extension], it becomes crucial that ESP implementation designed for constraint devices remain interoperable with the standard ESP implementation to avoid a fragmented usage of ESP. This document describes the the minimal properties and ESP implementation needs to meet.

For each field of the ESP packet represented in Figure 1 this document provides recommendations and guidance for minimal implementations. The primary purpose of Minimal ESP is to remain

interoperable with other nodes implementing RFC 4303 ESP, while limiting the standard complexity of the implementation.

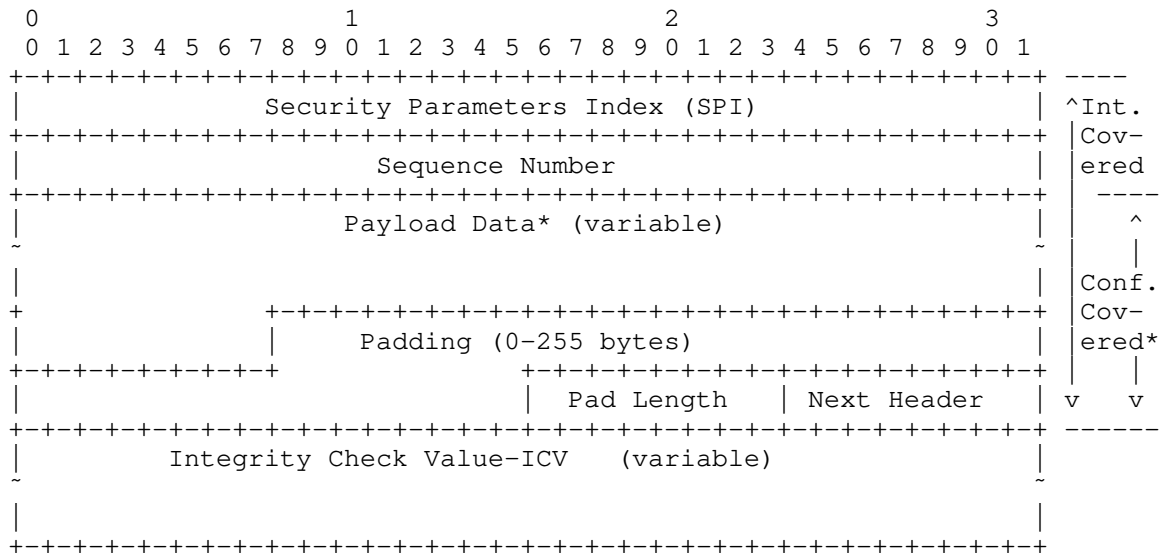


Figure 1: ESP Packet Description

3. Security Parameter Index (SPI) (32 bit)

According to the [RFC4303], the SPI is a mandatory 32 bits field and is not allowed to be removed.

The SPI has a local significance to index the Security Association (SA). From [RFC4301] section 4.1, nodes supporting only unicast communications can index their SA only using the SPI. On the other hand, nodes supporting multicast communications must also use the IP addresses and thus SA lookup needs to be performed using the longest match.

For nodes supporting only unicast communications, it is RECOMMENDED to index SA with the SPI only. Some other local constraints on the node may require a combination of the SPI as well as other parameters to index the SA.

It is RECOMMENDED to randomly generate the SPI indexing each inbound session. A random generation provides a stateless way to generate the SPIs, while keeping the probability of collision between SPIs relatively low. In case of collision, the SPI is simply re-generated.

However, for some constraint nodes, generating a random SPI may consume too much resource, in which case SPI can be generated using predictable functions or even a fix value. In fact, the SPI does not need to be random. Generating non random SPI MAY lead to privacy and security concerns. As a result, this alternative should be considered for devices that would be strongly impacted by the generation of a random SPI and after understanding the privacy and security impact of generating non random SPI.

When a constraint node uses fix value for SPIs, it imposes some limitations on the number of inbound SA. This limitation can be alleviated by how the SA lookup is performed. When fix SPI are used, it is RECOMMENDED the constraint node has as many SPI values as ESP session per host IP address, and that SA lookup includes the IP addresses.

Note that SPI value is used only for inbound traffic, as such the SPI negotiated with IKEv2 [RFC7296] or [RFC7815] by a peer, is the value used by the remote peer when it sends traffic. As SPI are only used for inbound traffic by the peer, this allows each peer to manage the set of SPIs used for its inbound traffic.

The use of fix SPI MUST NOT be considered as a way to avoid strong random generators. Such generator will be required in order to provide strong cryptographic protection and follow the randomness requirements for security described in [RFC4086]. Instead, the use of a fix SPI should only be considered as a way to overcome the resource limitations of the node, when this is feasible.

The use of a limited number of fix SPI or non random SPIs come with security or privacy drawbacks. Typically, a passive attacker may derive information such as the number of constraint devices connecting the remote peer, and in conjunction with data rate, the attacker may eventually determine the application the constraint device is associated to. If the SPI is fixed by a manufacturer or by some software application, the SPI may leak in an obvious way the type of sensor, the application involved or the model of the constraint device. When identification of the application or the hardware is associated to privacy, the SPI MUST be randomly generated. However, one needs to realize that in this case this is likely to be sufficient and a thorough privacy analysis is required. More specifically, traffic pattern MAY leak sufficient information in itself. In other words, privacy leakage is a complex and the use of random SPI is unlikely to be sufficient.

As the general recommendation is to randomly generate the SPI, constraint devices that will use a limited number of fix SPI are expected to be very constraint devices with very limited

capabilities, where the use of randomly generated SPI may prevent them to implement IPsec. In this case the ability to provision non random SPI enables these devices to secure their communications. These devices, due to there limitations, are expected to provide limited information and how the use of non random SPI impacts privacy requires further analysis. Typically temperature sensors, wind sensors, used outdoor do not leak privacy sensitive information. When used indoor, the privacy information is stored in the encrypted data and as such does not leak privacy.

As far as security is concerned, revealing the type of application or model of the constraint device could be used to identify the vulnerabilities the constraint device is subject to. This is especially sensitive for constraint devices where patches or software updates will be challenging to operate. As a result, these devices may remain vulnerable for relatively long period. In addition, predictable SPI enable an attacker to forge packets with a valid SPI. Such packet will not be rejected due to an SPI mismatch, but instead after the signature check which requires more resource and thus make DoS more efficient, especially for devices powered by batteries.

Values 0-255 SHOULD NOT be used. Values 1-255 are reserved and 0 is only allowed to be used internal and it MUST NOT be send on the wire.

[RFC4303] mentions :

"The SPI is an arbitrary 32-bit value that is used by a receiver to identify the SA to which an incoming packet is bound. The SPI field is mandatory. [...]"

"For a unicast SA, the SPI can be used by itself to specify an SA, or it may be used in conjunction with the IPsec protocol type (in this case ESP). Because the SPI value is generated by the receiver for a unicast SA, whether the value is sufficient to identify an SA by itself or whether it must be used in conjunction with the IPsec protocol value is a local matter. This mechanism for mapping inbound traffic to unicast SAs MUST be supported by all ESP implementations."

4. Sequence Number(SN) (32 bit)

According to [RFC4303], the Sequence Number (SN) is a mandatory 32 bits field in the packet.

The SN is set by the sender so the receiver can implement anti-replay protection. The SN is derived from any strictly increasing function that guarantees: if packet B is sent after packet A, then SN of packet B is strictly greater then the SN of packet A.

Some constraint devices may establish communication with specific devices, like a specific gateway, or nodes similar to them. As a result, the sender may know whereas the receiver implements anti-replay protection or not. Even though the sender may know the receiver does not implement anti replay protection, the sender **MUST** implement a always increasing function to generate the SN.

Usually, SN is generated by incrementing a counter for each packet sent. A constraint device may avoid maintaining this context and use another source that is known to always increase. Typically, constraint nodes using 802.15.4 Time Slotted Channel Hopping (TSCH), whose communication is heavily dependent on time, can take advantage of their clock to generate the SN. This would guarantee a strictly increasing function, and avoid storing any additional values or context related to the SN. When the use of a clock is considered, one should take care that packets associated to a given SA are not sent with the same time value.

For inbound traffic, it is **RECOMMENDED** to provide a anti-replay protection, and the size of the window depends on the ability of the network to deliver packet out of order. As a result, in environment where out of order packets is not possible the window size can be set to one. However, while **RECOMMENDED**, there is no requirements to implement an anti replay protection mechanism implemented by IPsec. A node **MAY** drop anti-replay protection provided by IPsec, and instead implement its own internal mechanism.

[RFC4303] mentions :

"This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. For a unicast SA or a single-sender multicast SA, the sender **MUST** increment this field for every transmitted packet. Sharing an SA among multiple senders is permitted, though generally not recommended. [...] The field is mandatory and **MUST** always be present even if the receiver does not elect to enable the anti-replay service for a specific SA."

5. Padding

The purpose of padding is to respect the 32 bit alignment of ESP. ESP **MUST** have at least one padding byte Pad Length that indicates the padding length. ESP padding bytes are generated by a succession of unsigned bytes starting with 1, 2, 3 with the last byte set to Pad Length, where Pad Length designates the length of the padding bytes.

Checking the padding structure is not mandatory, so the constraint device may not proceed to such checks, however, in order to

interoperate with existing ESP implementations, it MUST build the padding bytes as recommended by ESP.

In some situation the padding bytes may take a fix value. This would typically be the case when the Data Payload is of fix size.

[RFC4303] mentions :

"If Padding bytes are needed but the encryption algorithm does not specify the padding contents, then the following default processing MUST be used. The Padding bytes are initialized with a series of (unsigned, 1-byte) integer values. The first padding byte appended to the plaintext is numbered 1, with subsequent padding bytes making up a monotonically increasing sequence: 1, 2, 3, When this padding scheme is employed, the receiver SHOULD inspect the Padding field. (This scheme was selected because of its relative simplicity, ease of implementation in hardware, and because it offers limited protection against certain forms of "cut and paste" attacks in the absence of other integrity measures, if the receiver checks the padding values upon decryption.)"

ESP [RFC4303] also provides Traffic Flow Confidentiality (TFC) as a way to perform padding to hide traffic characteristics, which differs from respecting a 32 bit alignment. TFC is not mandatory and MUST be negotiated with the SA management protocol. TFC has not yet being widely adopted for standard ESP traffic. One possible reason is that it requires to shape the traffic according to one traffic pattern that needs to be maintained. This is likely to require extra processing as well as providing a "well recognized" traffic shape which could end up being counterproductive. As such TFC is not expected to be supported by a minimal ESP implementation.

As a result, TFC cannot not be enabled with minimal, and communication protection that were relying on TFC will be more sensitive to traffic shaping. This could expose the application as well as the devices used to a passive monitoring attacker. Such information could be used by the attacker in case a vulnerability is disclosed on the specific device. In addition, some application use - such as health applications - may also reveal important privacy oriented informations.

Some constraint nodes that have limited battery life time may also prefer avoiding sending extra padding bytes. However the same nodes may also be very specific to an application and device. As a result, they are also likely to be the main target for traffic shaping. In most cases, the payload carried by these nodes is quite small, and the standard padding mechanism may also be used as an alternative to TFC, with a sufficient trade off between the require energy to send

additional payload and the exposure to traffic shaping attacks. In addition, the information leaked by the traffic shaping may also be addressed by the application level. For example, it is preferred to have a sensor sending some information at regular time interval, rather when an specific event is happening. Typically a sensor monitoring the temperature, or a door is expected to send regularly the information - i.e. the temperature of the room or whether the door is closed or open) instead of only sending the information when the temperature has raised or when the door is being opened.

6. Next Header (8 bit)

According to [RFC4303], the Next Header is a mandatory 8 bits field in the packet. Next header is intended to specify the data contained in the payload as well as dummy packet. In addition, the Next Header may also carry an indication on how to process the packet [I-D.nikander-esp-beet-mode].

The ability to generate and receive dummy packet is required by [RFC4303]. For interoperability, it is RECOMMENDED a minimal ESP implementation discards dummy packets. Note that such recommendation only applies for nodes receiving packets, and that nodes designed to only send data may not implement this capability.

As the generation of dummy packets is subject to local management and based on a per-SA basis, a minimal ESP implementation may not generate such dummy packet. More especially, in constraint environment sending dummy packets may have too much impact on the device life time, and so may be avoided. On the other hand, constraint nodes may be dedicated to specific applications, in which case, traffic pattern may expose the application or the type of node. For these nodes, not sending dummy packet may have some privacy implication that needs to be measured. However, for the same reasons exposed in Section 5 traffic shaping at the IPsec layer may also introduce some traffic pattern, and on constraint devices the application is probably the most appropriated layer to limit the risk of leaking information by traffic shaping.

In some cases, devices are dedicated to a single application or a single transport protocol, in which case, the Next Header has a fix value.

Specific processing indications have not been standardized yet [I-D.nikander-esp-beet-mode] and is expected to result from an agreement between the peers. As a result, it is not expected to be part of a minimal implementation of ESP.

[RFC4303] mentions :

"The Next Header is a mandatory, 8-bit field that identifies the type of data contained in the Payload Data field, e.g., an IPv4 or IPv6 packet, or a next layer header and data. [...] the protocol value 59 (which means "no next header") MUST be used to designate a "dummy" packet. A transmitter MUST be capable of generating dummy packets marked with this value in the next protocol field, and a receiver MUST be prepared to discard such packets, without indicating an error."

7. ICV

The ICV depends on the crypto-suite used. Currently recommended [RFC8221] only recommend crypto-suites with an ICV which makes the ICV a mandatory field.

As detailed in Section 8 we recommend to use authentication, the ICV field is expected to be present that is to say with a size different from zero. This makes it a mandatory field which size is defined by the security recommendations only.

[RFC4303] mentions :

"The Integrity Check Value is a variable-length field computed over the ESP header, Payload, and ESP trailer fields. Implicit ESP trailer fields (integrity padding and high-order ESN bits, if applicable) are included in the ICV computation. The ICV field is optional. It is present only if the integrity service is selected and is provided by either a separate integrity algorithm or a combined mode algorithm that uses an ICV. The length of the field is specified by the integrity algorithm selected and associated with the SA. The integrity algorithm specification MUST specify the length of the ICV and the comparison rules and processing steps for validation."

8. Cryptographic Suites

The cryptographic suites implemented are an important component of ESP. The recommended suites to use are expected to evolve over time and implementer SHOULD follow the recommendations provided by [RFC8221] and updates. Recommendations are provided for standard nodes as well as constraint nodes.

This section lists some of the criteria that may be considered. The list is not expected to be exhaustive and may also evolve overtime. As a result, the list is provided as indicative:

1. Security: Security is the criteria that should be considered first for the selection of cipher suites. The security of cipher

suites is expected to evolve over time, and it is of primary importance to follow up-to-date security guidances and recommendations. The chosen cipher suites MUST NOT be known vulnerable or weak (see [RFC8221] for outdated ciphers). ESP can be used to authenticate only or to encrypt the communication. In the later case, authenticated encryption must always be considered [RFC8221].

2. **Interoperability:** Interoperability considers the cipher suites shared with the other nodes. Note that it is not because a cipher suite is widely deployed that is secured. As a result, security SHOULD NOT be weakened for interoperability. [RFC8221] and successors consider the life cycle of cipher suites sufficiently long to provide interoperability. Constraint devices may have limited interoperability requirements which makes possible to reduce the number of cipher suites to implement.
3. **Power Consumption and Cipher Suite Complexity:** Complexity of the cipher suite or the energy associated to it are especially considered when devices have limited resources or are using some batteries, in which case the battery determines the life of the device. The choice of a cryptographic function may consider re-using specific libraries or to take advantage of hardware acceleration provided by the device. For example if the device benefits from AES hardware modules and uses AES-CTR, it may prefer AUTH_AES-XCBC for its authentication. In addition, some devices may also embed radio modules with hardware acceleration for AES-CCM, in which case, this mode may be preferred.
4. **Power Consumption and Bandwidth Consumption:** Similarly to the cipher suite complexity, reducing the payload sent, may significantly reduce the energy consumption of the device. As a result, cipher suites with low overhead may be considered. To reduce the overall payload size one may for example:
 1. Use of counter-based ciphers without fixed block length (e.g. AES-CTR, or ChaCha20-Poly1305).
 2. Use of ciphers with capability of using implicit IVs [I-D.ietf-ipsecme-implicit-iv].
 3. Use of ciphers recommended for IoT [RFC8221].
 4. Avoid Padding by sending payload data which are aligned to the cipher block length - 2 for the ESP trailer.

9. IANA Considerations

There are no IANA consideration for this document.

10. Security Considerations

Security considerations are those of [RFC4303]. In addition, this document provided security recommendations and guidances over the implementation choices for each fields.

11. Acknowledgment

The authors would like to thank Daniel Palomares, Scott Fluhrer, Tero Kivinen, Valery Smyslov, Yoav Nir, Michael Richardson for their valuable comments.

12. References

12.1. Normative References

- [I-D.ietf-ipsecme-implicit-iv]
Migault, D., Guggemos, T., and Y. Nir, "Implicit IV for Counter-based Ciphers in Encapsulating Security Payload (ESP)", draft-ietf-ipsecme-implicit-iv-05 (work in progress), June 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.

- [RFC7815] Kivinen, T., "Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation", RFC 7815, DOI 10.17487/RFC7815, March 2016, <<https://www.rfc-editor.org/info/rfc7815>>.
- [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <<https://www.rfc-editor.org/info/rfc8221>>.

12.2. Informative References

- [I-D.mglt-ipsecme-diet-esp]
Migault, D., Guggemos, T., Bormann, C., and D. Schinazi, "ESP Header Compression and Diet-ESP", draft-mglt-ipsecme-diet-esp-06 (work in progress), May 2018.
- [I-D.mglt-ipsecme-ikev2-diet-esp-extension]
Migault, D., Guggemos, T., and D. Schinazi, "Internet Key Exchange version 2 (IKEv2) extension for the ESP Header Compression (EHC) Strategy", draft-mglt-ipsecme-ikev2-diet-esp-extension-01 (work in progress), June 2018.
- [I-D.nikander-esp-beet-mode]
Nikander, P. and J. Melen, "A Bound End-to-End Tunnel (BEET) mode for ESP", draft-nikander-esp-beet-mode-09 (work in progress), August 2008.

Appendix A. Document Change Log

[RFC Editor: This section is to be removed before publication]

-00: First version published.

-01: Clarified description

-02: Clarified description

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Email: daniel.migault@ericsson.com

Tobias Guggemos
LMU Munich
MNM-Team
Oettingenstr. 67
80538 Munich, Bavaria
Germany

Email: guggemos@mn-m-team.org