

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 2, 2018

M. Bagnulo
UC3M
Y. Nishida
GE Global Research
September 29, 2017

TCP ESN: Extended Sequence Numbers for TCP
draft-bagnulo-tcpm-esn-00.txt

Abstract

This note defines the Extended Sequence Number (ESN) experimental modification to TCP to increase TCP's sequence number using the TimeStamp (TS) option. It also modifies the Window Scale (WS) option to support larger receiver window enable by the extended sequence number space. At this stage, the purpose of this document is to discuss different design choices to generate discussion about the approach to follow.

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 2, 2018.

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 (<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

Internet-Draft

TCP ESN

September 2017

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.	Overview	2
2.	Design rationale	3
2.1.	Reduced option space consumption in the SYN and graceful fallback	4
2.2.	Deployability	4
3.	RTTM With Extended Sequence Number Prefix	4
4.	Middleboxes Implications	7
5.	SACK for Extended Sequence Number	8
6.	Impacts On Other TCP Extensions	8
6.1.	PAWS	8
6.2.	Eifel Detection Algorithm	9
7.	Acknowledgments	9
8.	Security Considerations	9
9.	IANA Considerations	9
10.	References	9
10.1.	Normative References	9
10.2.	Informative References	9
	Authors' Addresses	10

[1.](#) Overview

The proposed Extended Sequence Number (ESN) mechanism re-purposes the TS option [[RFC7323](#)] to carry a prefix for the sequence number and a prefix for the Acknowledgement number, increasing the sequence number used in TCP connections.

As currently defined, the TS option contains two 32-bit fields, TSval and TSecr. The current ESN proposal re-defines TSval to carry a prefix for the sequence number and TSecr to carry a prefix for the Acknowledgment number. In this way, the actual sequence number corresponding to the first data byte contained in the segment would be the concatenation of the value contained in the TSval and the value of the Sequence Number field of the TCP header. The Acknowledgment sequence number would be the concatenation of the value contained in the TSecr and the value of the Acknowledgment Number field of the TCP header.

The proposed ESN mechanism also modifies the WS option as follows: First, values up to 46 are allowed (enabling a RCV window up to 2^{46}). These are encoded in the 6 less significant bits of the shift.count. Second, the remaining two (most significant) bits are turned into flags. In particular, the most significant bit is used

as the ESN flag to indicate the ESN support in the connection. Specifically, when the ESN bit is set to 1 in the WS carried in a SYN or a SYN/ACK, it means that: i) the TS option is being used for extended sequence numbers, as defined above, and ii) that the sender of the WS option with the ESN bit set supports receiver window up to 2^{46} in this connection. The ESN flag defined this way allows endpoints to express and negotiate ESN support during the TCP 3-way handshake.

The sequence number of a TCP segment using ESN is the result of prepending the prefix carried in the TS Value and the sequence number contained in the Sequence Number field of the TCP header. Similarly, the ACK number is the result of prepending the value in the TS Echo Reply value and the value in the ACK field of the TCP header.

When a client wants to use the extended sequence number for a new connection, it sends a SYN with both the TS and the WS options. In the WS option, it sets the ESN flag to inform that it wants to use ESN for this connection. It encodes the most significant bits of the sequence number in the TS Value and the remaining bits of the extended sequence number in the sequence number field in the TCP header. Since the ACK flag is not set in the TCP header of the SYN packet, the TS Echo Value is set to zero (as defined in [RFC7323](#)).

If the server also supports the extended sequence number mechanism, the server replies with a SYN/ACK carrying both the TS and WS options. In the WS option it sets the ESN flag to confirm the ESN support. It encodes the prefix of its own extended sequence number in the TS Value and the prefix of the ACK in the TS Echo Reply.

If the server does not support ESN, it will respond with a SYN/ACK containing a WS option carrying a value lower than 14 i.e. with the most significant bit set to 0. It may also include the TS option indicating its willingness to use timestamps as defined in [RFC7323](#) in this connection. Upon the reception of the SYN/ACK, the client can gracefully fall back to use TS as defined in [RFC7323](#), in particular,

PAWS can be used.

2. Design rationale

Our proposal is to re-utilize the TCP TS option to carry a sequence number offset in addition to the existing 32 bits sequence number. This approach is similar to [[I-D.looney-tcpm-64-bit-segnos](#)] although it has distinct difference. while [[I-D.looney-tcpm-64-bit-segnos](#)] proposes to allocate a new TCP option, we propose to utilize existing TS option instead. We believe this approach will have the following advantages.

2.1. Reduced option space consumption in the SYN and graceful fallback

The maximum size of the TCP header (including options) is 60 bytes (this is because the Data Offset field of the TCP header is 4 bits and can express the offset in 32-bit words). Since the TCP basic header is 20 bytes, a segment can carry 40 bytes of options at most. This is particularly pressing for the TCP SYN and TCP SYN/ACK packets. Currently, there is a fair number of options that are frequently carried in SYN packets, especially in high performance communications. In particular, the MSS option (2 bytes) [[RFC0793](#)], the SACK permitted option (2 bytes) [[RFC2018](#)], the Window Scale option (3 bytes) and the TimeStamp option (used for PAWS) (10 bytes) [[RFC7323](#)]. All these options account for 17 bytes. There are other options that are becoming increasingly popular. For instance, The option length of TCP Fast Open (TFO) [[RFC7413](#)] is 6 bytes or 18 bytes depending on the length of the cookie used. There are other options that require SYN and SYN/ACK option space such as MP_CAPABLE in [[RFC6824](#)], or TCP-AO [[RFC5925](#)].

This means that for instance, a TCP client that would like to initiate a connection including the MSS option, SACK permitted option the WS and TS options and also carry a TFO option would not have room to carry an additional 10 byte long option for the extended sequence number. Since our approach utilizes TS option, additional option space for extended sequence number is not needed.

The proposed ESN approach allows for using the extended sequence number if both endpoints support it while enabling graceful fallback. A client supporting ESN would include the TS option and set

the flag in the WS option indicating the ESN support. If the server does not support ESN, the connection can still be established using 32 bit sequence numbers and the TS and WS options as defined in [RFC7323](#) (in particular PAWS can be used in the connection).

[2.2.](#) Deployability

[HONDA11] reported that unknown options in the SYN prone to be removed with higher probability than known options. Hence, we believe utilizing existing options will have better chances to avoid unwanted middleboxes' interferences. Although it would be useful to perform some other measurements specifically about how frequently the TS option is removed.

[3.](#) RTTM With Extended Sequence Number Prefix

[RFC7323] defined two uses for the TS option: PAWS and RTTM. When re-purposing the TS option for ESN, we argue that the use of TS for carrying extended sequence number subsumes the uses of PAWS.

However, this is not the case for RTTM. We identify the following alternatives in order to archive RTTM when re-purposing the TS option for ESN.

Option 1:

This approach uses the most significant bit (MSB) of both TSval and TSecr as a flag as depicted in Figure 1. If the MSB is set to 1, it means the field contained a sequence number prefix. If it is reset, it means that it contains a timestamp. This means that we use 31 bits for the extended sequence number prefix, resulting in 63 bit long sequence numbers. The main problem here is that the segments containing the timestamp lack the sequence number prefix information. So, for instance, it is not possible to have more than 2^{32} bytes in flight if any of the segments in flight is carrying an actual timestamp, since there is the possibility of confusion (in particular if the receive window is large enough to accommodate two packets with the same 32 bit sequence number, then the receiver would not be able to figure out the right place for the packet that carries the timestamp and does not carry the sequence number prefix). So, if we want to use this option, the receiver window cannot be larger than 2^{32} . However, this restriction does not address all the problems. If

a duplicated packet carrying a timestamp in the TS option gets delay one RTT or more and the 32 bit sequence number wraps around, then the receiver can potentially take this old duplicated packet for a new packet with the same sequence number suffix. It would be possible to rely on PAWS for detecting and eliminating this packets. However, in order for PAWS to be used, it is necessary to keep the timestamp information stored in TS.recent updated. This requires that at least a few actual timestamps are exchanged every 2^{31} sequence numbers. Summarizing, the constraints to use this option are first that the light-size is less than 2^{32} and that at least n ($n=4?$) timestamps are exchanged every 2^{32} bytes of data. We believe this is poor alternative, especially due to the flight-size constraint.

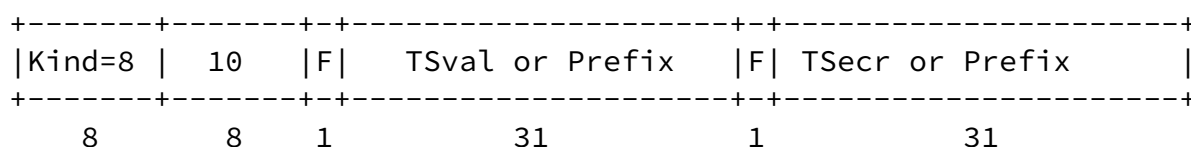


Figure 1: Time Stamp Option format for Option 1

Option 2:

This approach uses the TSecr in some packets to exchange timestamps. The idea here is that all data segments carry the extended sequence number prefix in the TSval but that some packets do not carry ACK information, which is acceptable because we use cumulative ACKs as long as this only affects a few packets (e.g. one packet per RTT do not carry ACK information). In order to enable both uses of the TSecr (timestamp or sequence number prefix), we need to use 2 bits to encode whether the TSecr carries either an extended sequence number prefix for the ACK, a timestamp or a timestamp echo. This implies that there are 30 bits left in TSecr for the actual value, resulting in 30 bit timestamps and 62 bit sequence numbers. The receiver of a packet carrying the TS option carrying an actual timestamp or timestamp echo should discard the ACK information since it cannot know the the prefix of the seq number carried in the ACK field. This

option seems a reasonable trade-off. If this option is adopted, RTTM could only be used sporadically. However, this may not be a concern, since it is likely that it would be possible to measure the RTT at least once every RTT which is likely to be enough for estimating the RTT for the RT0 calculation (see [[RFC7323](#)] for further details).

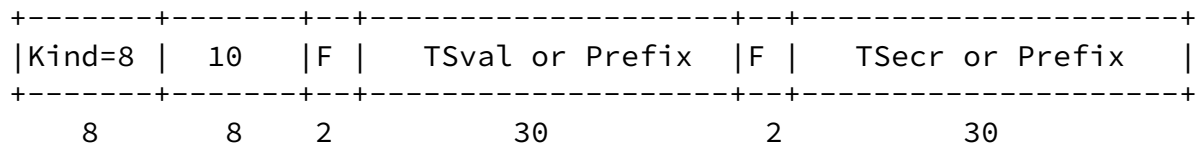


Figure 2: Time Stamp Option format for Option 2

Option 3:

This approach splits the TSval and the TSecr into two 16-bit fields resulting in 16 bit timestamps and 48 bit sequence numbers. 48 bit sequence numbers are a significant improvement from the current 32 bit sequence numbers, so it is probably enough. It is possible to encode the timestamp information using 16 bits. For example, [[I-D.trammell-tcpm-timestamp-interval](#)] proposes to encode timestamp information using 16 bits, which could be used in this option.

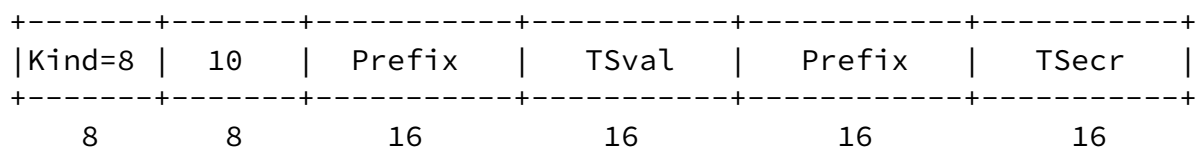


Figure 3: Time Stamp Option format for Option 3

Option 4:

This approach Only uses the TS for one single purpose per connection either the original purpose or ESN. This will be less attractive because the RTTM cannot be used with ESN in the same connection.

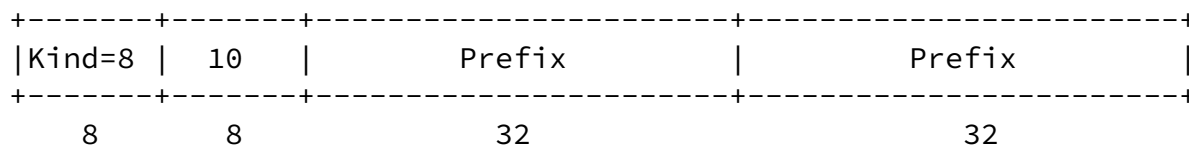


Figure 4: Time Stamp Option format for Option 4

Based on the observations above, we believe option 2 and 3 would be worth for further discussions while option 1 and 4 can be discarded due to major drawbacks.

4. Middleboxes Implications

It has been observed in [[HONDA11](#)] that some middleboxes insert the TS Option. Also, there may be boxes out there that modify the sequence number, while not terminating the connection. In order to detect these cases that would break the proposed mechanism, it would be beneficial to add an extra safety measure requiring that the prefix encoded in the TS Option replicates the most significant bits of the value included in the Sequence number field. In this way, a server supporting the extended sequence number mechanism cannot only verify the flag in the WS option, but also check if the TS value matches with the 31 most significant bits in the Sequence Number field in the TCP header. If they do not match, the server should not negotiate the use of the extended sequence number mechanism (i.e. it replies with the WS option resetting the flag for the extended sequence number mechanism). This is adopted from [[I-D.looney-tcpm-64-bit-seqnos](#)].

In case that the server is a legacy server, it will reply without the WS option or with the WS option with a shift.count value lower than

15. In this case, the client falls back to regular TCP without the

extended sequence number and regular timestamps.

5. SACK for Extended Sequence Number

In the case of SACK blocks, there are two possible complementary approaches:

1. we use the currently defined SACK options identifying bits using 32 bit sequence numbers. These are used in a connection that has successfully negotiated ESN, the prefix carried in the TSecr of the message applies also to the sequence numbers identifying the SACK blocks. The limitation of such approach is that all SACK blocks in a single SACK option must use to the same prefix, which prevents from SACKing older blocks. However, it is not certain that if we really need to report wide range of SACK blocks in a single SACK option. Another issue would be the case where a SACK option is detached from the original packet and attached to a different one. One possible mitigation for this would be discarding SACK info in case of suspicious as SACK is optional info and a SACK info usually is carried in multiple ACKs.
2. define a new SACK block option for extended sequence numbers as proposed in [[I-D.looney-tcpm-64-bit-seqnos](#)].

There are a couple of observations regarding the last option using the new SACK block option. First, note that the currently SACK permitted option could still be used. Hence, if a connection negotiated both SACK and ESN, we may presume that it supports the new SACK block option. If the ESN negotiation fails, it means that 32-bit SACK are to be used for that connection, providing graceful fallback.

6. Impacts On Other TCP Extensions

Since this proposal repurpose the existing use of timestamp option, some other proposals that use the option will be affected. We investigated the impacts on the following TCP extensions and propose modifications to make them work with the proposal.

6.1. PAWS

In order to perform PAWS, receives need to check if the timestamp option in an arrived packet contains sequence number prefix or timestamp info by checking the most significant bit. If it contains timestamp info, it process the timestamp info as described [Section 5.3 in \[RFC7323\]](#). If it contains sequence number prefix, it can know the extended sequence number of the packet based on the

into. If the extended sequence number is outside of the window, the packet will be discarded as PAWS.

[6.2.](#) Eifel Detection Algorithm

If Eifel detection algorithm [[RFC3522](#)] is activated, senders performs the logics described in [Section 3.2 of \[RFC3522\]](#) with the following two modifications. First, TCP sender MUST set timestamp info when it retransmit packets. Second, if TCP sender receives the ACK with sequence number prefix for the retransmitted packet, it should treat as if the timestamp is smaller than the value of RetransmitTS.

[7.](#) Acknowledgments

[8.](#) Security Considerations

[9.](#) IANA Considerations

[10.](#) References

[10.1.](#) Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", [RFC 7323](#), DOI 10.17487/RFC7323, September 2014, <<https://www.rfc-editor.org/info/rfc7323>>.

[10.2.](#) Informative References

- [HONDA11] Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., and H. Tokuda, "Is it still possible to extend TCP?", ACM IMC 2011, 2011.
- [I-D.looney-tcpm-64-bit-seqnos] jlooney@juniper.net, j., "64-bit Sequence Numbers for TCP", [draft-looney-tcpm-64-bit-seqnos-00](#) (work in progress), March 2017.
- [I-D.trammell-tcpm-timestamp-interval] Scheffenegger, R., Kuehlewind, M., and B. Trammell, "Encoding of Time Intervals for the TCP Timestamp Option", [draft-trammell-tcpm-timestamp-interval-01](#) (work in progress), March 2017.

progress), July 2013.

Internet-Draft

TCP ESN

September 2017

- [RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", [RFC 2018](#), DOI 10.17487/RFC2018, October 1996, <<https://www.rfc-editor.org/info/rfc2018>>.
- [RFC3522] Ludwig, R. and M. Meyer, "The Eifel Detection Algorithm for TCP", [RFC 3522](#), DOI 10.17487/RFC3522, April 2003, <<https://www.rfc-editor.org/info/rfc3522>>.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", [RFC 5925](#), DOI 10.17487/RFC5925, June 2010, <<https://www.rfc-editor.org/info/rfc5925>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", [RFC 6824](#), DOI 10.17487/RFC6824, January 2013, <<https://www.rfc-editor.org/info/rfc6824>>.
- [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", [RFC 7413](#), DOI 10.17487/RFC7413, December 2014, <<https://www.rfc-editor.org/info/rfc7413>>.

Authors' Addresses

Marcelo Bagnulo
UC3M

Email: marcelo@it.uc3m.es

Yoshifumi Nishida
GE Global Research
2623 Camino Ramon
San Ramon, CA 94583
USA

Email: nishida@wide.ad.jp

