Network Working Group
Internet-Draft
Intended status: Experimental
Expires: August 3, 2017

Y. Nishida
GE Global Research
H. Asai
The University of Tokyo
M. Bagnulo
UC3M

January 30, 2017

Increasing Maximum Window Size of TCP
draft-nishida-tcpm-maxwin-03.txt

Abstract

This document proposes to increase the current max window size allowed in TCP. It describes the current logic that limits the maximum window size and provides a rationale to relax the limitation as well as the negotiation mechanism to enable this feature safely.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

TCP throughput is determined by two factors: Round Trip Time and Receive Window size. It can never exceed Receive Window size divided by RTT. This implies larger window size is important to achieve better performance. Original TCP’s maximum window size defined in RFC793 [RFC0793] is 2^16 -1 (65,535), however, RFC7323 [RFC7323] defines TCP Window Scale option which allows TCP to use larger window size. Window Scale uses a shift count stored in 1-byte field in the option. The receiver of the option uses left-shifted window size value by the shift count as actual window size. When Window Scale is used, TCP can extend maximum window size to 2^30 - 2^14 (1,073,725,440). This is because the maximum shift count is 14 as described in the Section 2.3 of RFC7323 [RFC7323]. However, since TCP’s sequence number space is 2^32, we believe it is still possible to use larger window size than this while careful design of the logic that can identify segments inside the window is required. In this document, we propose to increase the maximum shift count to 15, which extend window size to 2^31 - 2^15.

2. Conventions 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 [RFC2119].

3. Increasing Maximum Window Size

RFC7323 requires maximum window size to be less than 2^30 as described below.

"

TCP determines if a data segment is "old" or "new" by testing whether its sequence number is within 2^31 bytes of the left edge of the window, and if it is not, discarding the data as "old". To insure that new data is never mistakenly considered old and vice versa, the left edge of the sender’s window has to be at most 2^31 away from the right edge of the receiver’s window. The same is true of the sender’s right edge and receiver’s left edge. Since the right and left edges of either the sender’s or receiver’s window differ by the window size, and since the sender and receiver windows can be out of phase by at most the window size, the above constraints imply that two times the maximum window size must be less than 2^31, or

max window < 2^30

"

However, TCP does not necessarily need to determine if a segment is old or new. Because important point is to determine if a receive segment is inside of the window or not. It basically does not matter if a segment is too old (left side of the window) or too new (right side of the window) as long as it is outside of the window. Based on this viewpoint, we propose to extend maximum window to 2^31 - 2^15, which can be attained by increasing maximum shift count to 15.

To demonstrate the feasibility of the proposal, we would like to use the following worst case example where the sender and the receiver windows are completely out of phase. In this example, we define S as the sender’s left edge of the window and W as the sender’s window size. Hence, the sender’s right edge of the window is S+W. Also, the receiver’s left edge of the window is S+W+1 and the right edge of the window is S+2W+1, as they are out of phase. This situation can happen when the sender sent all segments in the window and the receiver received all segments while no ACK has been received by the sender yet. Now, we presume a segment that contains sequence number S has arrived at the receiver. This segment should be excluded by the receiver, although it can easily happen when the sender retransmits segments.

In case of W=2^31, the receiver cannot exclude this segment as S+2W = S. It is considered inside of the window. (S+W+1 < S < S+2W+1) However, our proposed window size is W=2^31-X, where X is 2^15. In this case, when segment S has arrived, the following checks will be performed. First, TCP checks it with the left edge of the window and it considers the segment is left side of the left edge. (S < S+W+1 Note: W=2^31-X) Second, TCP checks it with the right edge of the window and it considers the segment is right of the right edge. (S > S+2W+1) You might notice that the result of the second check is not expected one as the segment S is actually an old segment. This is the problem that the referred paragraphs from RFC7323 [RFC7323] describe. However, the segment is properly excluded by the receiver as both checks indicate it is outside of the window. It should be noted that the principle of TCP requires to accept the segment S only when it has passed both checks successfully, which means S must satisfy the following condition.

```
S >= left edge && S <= right edge
```

As we have shown in the example, our proposed maximum window size: W=2^31-2^15 does not affect this principle.

Using the larger window size implies that the sequence number space can wrap around in less than 3 RTTs. This can pose problems to distinguish old retransmitted packets from new packets solely using the same sequence number. Because of this, a sender using the larger window size defined in this specification is recommended to use Protection Against Wrapped Sequences (PAWS) as defined in RFC7323 [RFC7323].

4. Updating the Window Scale Option

As shown in Figure 1, the Window Scale Option (WSO) defined in [RFC7323] has three 1-byte fields, the Kind field (which specifies the option type), the Length field (set to 3 because the WSO is 3 bytes long) and the shift.cnt field (which specifies the shift count applied to the window to scale it).

```
+-----------+-----------+-----------+
|   Kind=3  |  Length=3 | shift.cnt |
+-----------+-----------+-----------+
      1           1          1
```

Figure 1: Window Scale Option (WSO) format

RFC7323 [RFC7323] defines that the shift.cnt field can have a maximum value of 14 and upon reception of a larger value in this field, the receiver must proceed as if it had received a shift.cnt of 14.

This specification updates the shift.cnt field definition. Figure 2 represents the new format of the shift.cnt field. The eight bits contained in the shift.cnt field are formatted as "SSSSLRRR".

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|S S S S L R R R|
+-+-+-+-+-+-+-+-+
```

Figure 2: New shift.cnt field format

These bits are parsed as follows:

- The four leftmost bits "SSSS" express the shift-count, as in RFC7323 [RFC7323], only that now the maximum shift count value allowed is 15.

- The "L" bit expresses if the sender supports the large window defined in this specification i.e. the bit is set if the sender supports this specification.

- The three rightmost bits "RRR" are reserved for future use and MUST be set to zero.

This new format for the shift.count field allows an updated client to initiate a TCP connection and express that it supports the larger window by setting the "L" bit, while still conveying information about the shift count that it wants to use for its own RCV.WND in the four leftmost bits "SSSS" (which do not necessarily have to be set to 15). A server that supports this specification that receives a SYN with the WSO with the "L" bit set knows that it can reply using a shift count of 15. A legacy server that receives the WSO with the "L"" bit set will interpret it using the RFC7323 format and will then read it as a shift count value larger than 14. As per RFC7323 the server MUST then assume a shift count of 14. The legacy server will then reply with a WSO with the "L" bit set to zero, so the client knows that the server does not support this specification and that the server will assume a shift count of 14 for the client’s receive window.

5. Use Cases, Benefits to Explore Maximum Window Size

One of the use cases of the extended maximum window size is high volume data transfer over paths with long RTT delays and high bandwidth, called long fat pipes. The proposed extension improves and doubles at most the maximum throughput when bandwidth-latency product is greater than 1 GB. As propagation delay in an optical fiber is around 20 cm/ns, RTT will be over 100 milliseconds when the distance of the transmission is more than 10000km. This distance is not extraordinary for trans-pacific communications. In this case, the maximum throughput will be limited to 80 Gbps with the current maximum window size, although network technologies for more than 100 Gbps are becoming common these days.

As the current TCP sequence number space is limited to 32 bits, it will not be possible to increase maximum window size any further. However, TCP may eventually have other extensions to increase sequence number space, for example, [RFC7323] and [RFC1263] mention about increasing sequence number space to 64 bits. We believe the information in this document will be useful when such extensions are proposed as they need to define new maximum window size.

6. Acknowledgments

The authors gratefully acknowledge significant inputs for this document from Richard Scheffenegger and Ilpo Jarvinen.

7. Security Considerations

It is known that an attacker can have more chances to insert forged packets into a TCP connection when large window size is used. This is not a specific problem of this proposal, but a generic problem to use larger window. Using PAWS can mitigate this problem, however, it is recommended to consult the Security Considerations section of RFC7323 [RFC7323] to check its implications.

8. IANA Considerations

If approved, this document overrides the definition of the WSO option defined in RFC7323 and so the IANA registry should be update accordingly (at least to add a pointer to this specification as reference for the WSO in the IANA registry).

9. References

9.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[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>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <http://www.rfc-editor.org/info/rfc7323>.

9.2. Informative References

[RFC1263] O’Malley, S. and L. Peterson, "TCP Extensions Considered Harmful", RFC 1263, DOI 10.17487/RFC1263, October 1991, <http://www.rfc-editor.org/info/rfc1263>.

Authors’ Addresses

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

Email: nishida@wide.ad.jp

Hirochika Asai
The University of Tokyo
7-3-1 Hongo
Bunkyo-ku, Tokyo 113-8656
JP

Email: panda@wide.ad.jp

Marcelo Bagnulo
UC3M

Email: marcelo@it.uc3m.es