Workgroup: TCPM Working Group
Internet-Draft:
draft-thejeswara-tcpm-tcp-fwa-00
Published: 8 February 2023
Intended Status: Informational
Expires: 12 August 2023
Authors: R. Thejeswara Samsung
H. Kothari Samsung
S. Chinthalapudi Samsung

TCP FWA: Fast Window Advance for TCP

Abstract

This document describes TCP Fast Window Advance (FWA) flag in TCP Header to avoid the Head of Line Blocking in TCP long alive connections and in TCP long fat networks. FWA flag shall be set by the sender to force the TCP receiver to change its expected sequence number. This allows the application sender to send fresh application session to receiver on the existing TCP connection without blocking on the earlier session. The FWA flag is will solve long standing Head of Line (HOL) blocking problem in TCP for certain TCP applications.

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 12 August 2023.

Copyright Notice

respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Transmission Control Protocol (TCP) is described in [RFC9293] provides the robust and reliable transport layer protocol. TCP recovers data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted by sender. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates and error detection is performed by using checksum. Any loss of segments or error segments requires at retransmission by TCP sender.

TCP receiver uses the ACK field to acknowledge the sequence numbers it has received and can optionally use the SACK option to report the sequence numbers received in out of order.

Certain applications such as 3GPP VoLTE/VoNR(Voice over LTE/Voice over NR) have SIP session timers over TCP connections to wait for response from server. Once SIP Session timeout occurs, application treats that session as failure and continues with next session. At this point, TCP is unaware of session failure and it continues to retransmit unacknowledged segments of previous SIP session, and IMS

application on receiver side receives the stale SIP session and replies to session.

Once the SIP session is treated as failed, the IMS application generate the new SIP session, the new SIP session will have different session tokens from previous session, so the IMS application will reject any further responses to old SIP session from the receiver. SIP receiver sends the response to old sessions, as a result IMS/SIP sender either terminates the current session or has to ignore the old SIP response messages

As a result, in intermit network conditions new SIP session data is either buffered or failed with TCP HOL blocking. As in [IMS] architecture TCP data buffering can occur at multiple intermediaries in the end-to-end network topology of IMS Network.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Definitions

We repeat here some of the definitions from [RFC5681] to aid the reader.

SENDER MAXIMUM SEGMENT SIZE (SMSS): The SMSS is the size of the largest segment that the sender can transmit. This value can be based on the maximum transmission unit of the network, the path MTU discovery [RFC1191], [RFC4821] algorithm, RMSS (see next item), or other factors. The size does not include the TCP/IP headers and options.

RECEIVER MAXIMUM SEGMENT SIZE (RMSS): The RMSS is the size of the largest segment the receiver is willing to accept. This is the value specified in the MSS option sent by the receiver during connection startup. Or, if the MSS option is not used, it is 536 bytes [RFC1122]. The size does not include the TCP/IP headers and options.

RECEIVER WINDOW (rwnd): The most recently advertised receiver window.

CONGESTION WINDOW (cwnd): A TCP state variable that limits the amount of data a TCP can send. At any given time, a TCP MUST NOT send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of cwnd and rwnd.

Head of Line Blocking (HOL) : A protocol state TCP can not process the received data, until the missing segment is received.

4. Session Failure Problem

4.1. IMS Session

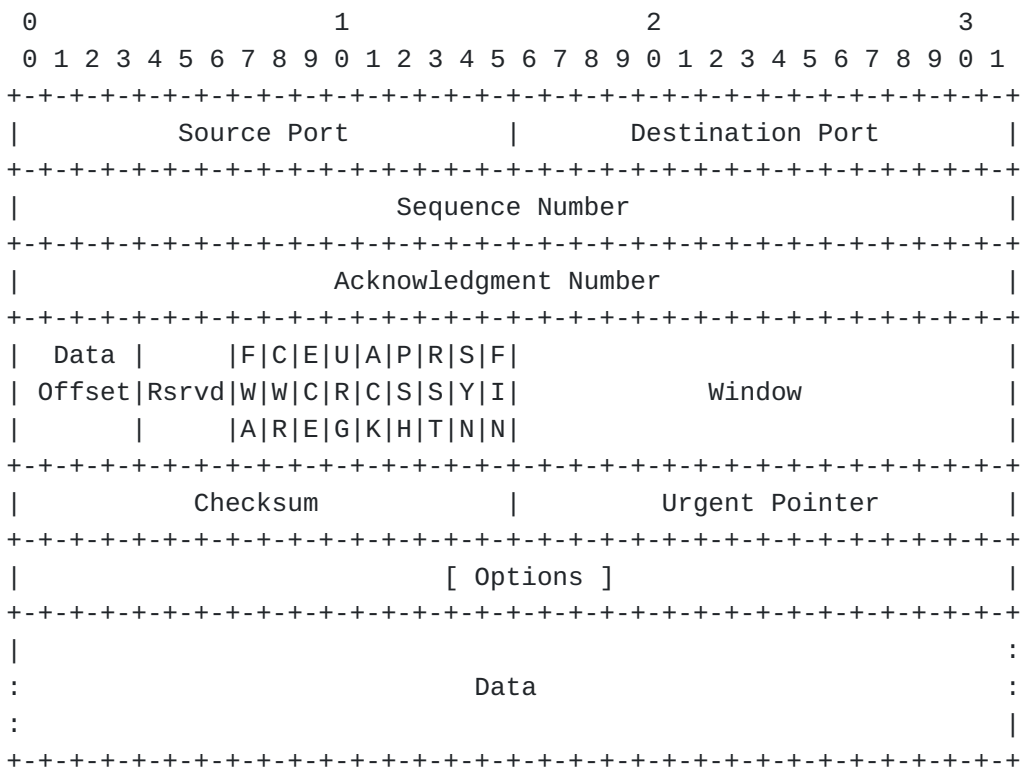
In IMS based applications, SIP Signalling [RFC3261] is performed using TCP protocol.When user triggers a voice call, SIP INVITE is sent to peer and IMS Stack waits for 1xx response. SIP INVITE packet size is in multiple of TCP Maximum Segment size (MSS) and required TCP level segmentation based on the negotiated SMSS with TCP peer RMSS during 3-way handshake.This has been observed from field test that sometimes complete TCP segments carrying SIP INVITE has not reached TCP peer and TCP peer is still waiting for remaining data to pass SIP INVITE message to SIP decoder. SIP Decoder needs complete SIP Payload to process the SIP transaction. Due to intermit network conditions, these remaining TCP segments are permanently lost and/or unordered. These received TCP segments received at IMS receiver after the Sender IMS Session/transaction timeout.

SIP Call originator [IMS] has shorter call setup timer duration (example: 6 seconds, Call setup timer is configurable by cellular network operator) in comparison to overall TCP retransmission attempts time out(example : 60 seconds) to conclude call attempt as failure. In such cases call originator has concluded call attempt as failure based on call setup timer and new SIP/IMS call transaction is made for retry of session. At this stage, TCP peer/receiver is still waiting for remaining part of previous SIP INVITE(SIP session packets). This leads to session failure or delay for all further sessions until TCP connection is reset and re-connected or pending data is received and acknowledged by the receiver.

The existing methods addresses this problem by changing congestion window (cwnd) of the sender based on congestion control algorithms example : [RFC8312]. A paper on HOL [OPTHOL] addresses the HOL blocking by using Forward Error Correction (FEC). But congestion control algorithms and FEC cannot solve the issue of HOL blocking, until the transmitted segments are acknowledged by the TCP receiver or recovered by using FEC. The delayed acknowledgment is causes multiple session failures when packet loss probability is high and further application transaction fails due to delay in application session response.

5. TCP Header Modification

This paper proposes to extend TCP flags by utilizing one bit among the four reserved bits. This new control bit is termed as Fast Window Advance (FWA) as shown in below figure



Note that one tick mark represents one bit position.

Figure 1: Modified TCP Header Format

where control bit

FWA : 1 bit

Fast Window Update

When application has closed its session as failure and application wants TCP layer of both sender and receiver to flush any previous data even if application data is not acknowledged by the receiver.

application can configure setting FWA flag through socket option interface.

6. FWA Operation

TCP control block in the sender and receiver will store the Sequence Number Variables related to the TCP connection such as

SND.NXT - send next (sequence number to be used to send next data)

RCV.NXT - receive next (expected sequence number at the receiver)

as described in [RFC9293].

This document adds one such variable (SND.FWS) to TCP control block

SND.FWS - send fast window to store the sequence number to be used in FWA bit (1) Segment

SND.FWS is set to initial value of ISS (initial send sequence number) during the connection establishment. When the TCP Sender Sends the data to receiver. SND.NXT is updated with next in sequence sequence number. so

if SND.FWS < SND.NXT then

   <SND.FWS=SND.NXT>

SND.FWS stores maximum sequence number sent by the TCP in the connection.

When the socket option to set FWA flag is received from the application TCP sender will set the FWA Flag in the next in-sequence segment to the receiver. TCP Sender use sequence number as in SND.FWS in FWA(1) TCP Segment. The sender application logic choses to set TCP FWA flag to discard any pending previous data at the TCP buffers. Through this socket option TCP sender invokes the FWA functionality.

<SEQ=SND.FWS><ACK=RCV.NXT><CTL=FWA,ACK>

At the TCP receiver, the FWA flag will force to advance its receiver window to set RCV.NXT to same as received sequence number and clear any pending data present in the receive buffer up to SND.FWS.

if (FWA is 1 in TCP FLAGS)

   <RCV.NXT=SEG.SEQ>

TCP receiver, will acknowledge the change in the RCV.NXT through ACK segment to the sender. TCP sender uses the TCP ACK segment to flush the pending buffer in the sender's buffer up to the sequence number (RCV.NXT) received in the TCP ACK segment.

Through this mode of operation, TCP receiver window will avoid HOL blocking of previously timed out/failed transaction and sends ACK to sender with updated sequence in ACK field of TCP header. This will allow both sender and receiver to continue next session without any HOL blocking caused by previous failed application session. TCP Header with FWA flag operation can clear partial received data that is waiting for remaining segments.

6.1. FWA behaviour

There is no negotiation needed in TCP handshake to negotiate the support of TCP FWA, TCP implementations which does not support FWA interpretation of TCP header can continue by ignoring the FWA bit. Since the flushing and changing the sequence number depends on positive ACK from the receiver. if the receiver does not support FWA, receiver sends the next expected sequence number as per the [RFC9293], so sender can not advance the window.

The receiver after receiving the segment with FWA bit set , should not delay the sending of the ACK even though delayed ACK is enabled and receiver SHALL repeat sending the ACK every 0.5 secs until sequence number from the sender changes from previous SND.FWS sequence number. This is to accelerate the buffer cleaning procedure in the sender and middle boxes.

The FWA bit does not impact the regular TCP connections which does not implement TCP Window Advance (FWA bit) both on sender and receiver.

6.1.1. Impact on Middle Boxes

There is no impact on stateless middle boxes, as these middle boxes are transparent or not alter the sequence numbers in between the end points. In stateful middle boxes, shall use the sequence numbers mapping between the entities to map the received sequence number,to generate the new sequence number of outgoing TCP connections.

6.1.2. Impact on Good put

Addition of FWA flag based on senders discretion improves the good put of the TCP. As the unnecessary retransmissions for timed out applications sessions are avoided.

7. Security Considerations

This document defines a a new flag in TCP Header which do not add any new security concerns beyond those discussed in [RFC9293].

8. IANA Considerations

This document requests new TCP flag in TCP header to indicate the FWA bit.

9. References

9.1. Normative References

[IMS]

3GPP, TS 24.229 V18.0.0., "IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP);Stage 3(Release 18)", 2020, <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1055>.

[OPTHOL] Mehrotra, S., "An Optimal Solution to Head-of-Line Blocking", 2020, <https://www.microsoft.com/en-us/research/uploads/prod/2020/02/paper.pdf>.

[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>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

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

[RFC8312] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", RFC 8312, DOI 10.17487/RFC8312, February 2018, <https://www.rfc-editor.org/info/rfc8312>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

9.2. Informative References

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

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC8174]
Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Authors' Addresses

Thejeswara Reddy
Samsung
Outer Ring Road
Bangalore 560048
Karnataka
India

Email: thejeswar.p@samsung.com

Harsh Kothari
Samsung
Outer Ring Road
Bangalore 560048
Karnataka
India

Email: harsh.mk@samsung.com

Srinivas Chinthalapudi
Samsung
Outer Ring Road
Bangalore 560048
Karnataka
India

Email: srinivas.y84@samsung.com