TCP Extended Acknowledgment Options
draft-huang-tsvwg-tcp-eack-00.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 28, 2010.

Copyright Notice

Copyright (c) 2010 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 BSD License.
Abstract

TCP may experience asymmetric bi-directional performance and poor over all performance when acknowledgments (ACKs) are sent by separate packets instead of being piggybacked by data packets. Between two TCP end points, when two-way traffic flows do not share the same TCP connection, the increase of traffic volume in one direction consumes more bandwidth in the reverse direction by generating more separate ACK packets, potentially causing more congestions in the reverse direction.

An Extended Acknowledgment (EACK) mechanism can help overcome these limitations. The receiving TCP sends back EACK packets to the sender informing the sender of data, including data of the current TCP connection and data of other eligible connections, that have been received. EACK thus reduces the need for separate ACK packets and improves the bi-directional symmetry and over all throughput.

This memo proposes an implementation of EACK and discusses its related issues.

Acknowledgements

Some of the text in this document is taken directly from RFC 2018 "TCP Selective Acknowledgement Options" by Mathis, et. al [Mat96]. The authors would like to thank Jiebing Wang for his review and constructive comments.

1. Introduction

TCP uses the piggyback ACK mechanishm to reduce the overhead of the TCP ACK transmissions. This mechanism is effective if the TCP connection carries bi-directional traffic. When multiple connections are established and each carries one-way traffic, even if the over all traffic is bi-directional, a TCP receiver can not piggyback ACKs on any data packets but have to generate separate ACK packets. With the delayed ACK algorithm [Bra89], a TCP receiver still SHOULD generate an ACK packet for at least every second full-sized segment [All99]. Generating separate ACK packets potentially has two performance issues.

The first issue is the unstablility of the TCP performance. For the ease of discussion, a specific case is considered with the following assumptions:
- A and B are two TCP hosts;
- A sends data to B on a connection while B sends data to A on another connection;
- both A and B always have full-sized segments ready for send; and
- the available bandwidth is symmetric for the two directions.

The unstability cycle starts at the point A and B are sending data traffic at the same rate, and the ACK packets to data packets ratio is already the lowest at 1:2. Some time later, however, there is an increase of data traffic volume in A->B direction. The increase of A->B data traffic requires more separate ACK packets in B->A direction, thus consumes more bandwidth and creates more congestions in B->A direction. If congestions result in packet drop in the B->A direction, more data packets than ACK packets are dropped. The loss of an ACK packet generally has less impact on the TCP throughput because the next ACK is likely to cover the previous ACK, however, each loss of data packet will result in a reduction of slow start threshold and congestion window [All99]. The combined result is a positive feedback that keeps exaggerating the asymmetric behavior until other limiting factors kick in.

The second issue is the reduced total TCP throughput. The separate ACK packets certainly create an overhead for the TCP transmission. This overhead is magnified when IP payload compression (IPcomp) is applied. Using IPcomp, because data packets in many cases are more compressible than the small separate ACK packets, the portion of bandwidth consumed by separate ACK packets is increased. The symmetric behavior further reduces the total throughput as the gain of performance on the faster direction can not make up the loss on the slower direction.

FTP always sends one-way traffic at a time on the data connection. Using FTP, bi-directional performace testing can be done by running FTP client simutanuously on both hosts. TCP performance test tool Iperf [Iperf] also sends one-way traffic on each connection, and create two connections for bi-directional traffic. These two issues have been demonstrated by running FTP or Iperf on two hosts with different CPU power. In contrast, another TCP performance testing tool NetStrain [Netst] sends two-way traffic over the same connection, thus these two issues are not seen by running NetStrain.

Extended Acknowledgment (EACK) is a strategy which mitigates these issues in the face of multiple connections each of which carries one way traffic. With extended acknowledgments, the data receiver can use one packet to acknowledge the arrival of segments of multiple connections sharing the same source and destination internet addresses, thus the chance to piggyback an ACK instead of sending a separate ACK packet is greatly improved.

The extended acknowledgment extension uses two TCP options. The first is an enabling option, "EACK-Permitted", which may be sent in
a SYN segment to indicate that the EACK option can be used on the connection once it is established. The other is the EACK option itself, which may be sent over an established connection once permission has been given by EACK-Permitted. Using the EACK option on a connection means packets carrying EACK option can be sent on that connection, and the acknowledgement of that connection can also be sent by EACK option in packets of other proper connections that are EACK-permitted.

The EACK option is to be included in a segment sent from a TCP that is receiving data to the TCP that is sending that data; these TCP’s are referred to as the data receiver and the data sender, respectively. A particular simplex data flow is considered in the discussion; any data flowing in the reverse direction can be treated independently.

2. Internet Connection Group

All TCP connections sharing the same source and destination internet addresses form a group. This group represent all TCP connections between two end points of an internet connection, and is referred to as an internet connection group. EACK option extends the acknowledgement coverage from a single connection to multiple connections belong to an internet connection group.

3. EACK-Permitted Option

This two-byte option may be sent in a SYN by a TCP that has been extended to receive (and presumably process) the EACK option. By sending the EACK-Permitted option, the data sender advertizes to the data receiver that,

1. the data receiver can use the intended connection to carry an EACK option, and
2. the data receiver can send the acknowledgement for the intended connection in an EACK option carried by another connection.

This option MUST NOT be sent on non-SYN segments.

TCP Sack-Permitted Option:

Kind: 29

+--------+--------+
| Kind=29 | Length=2|
+--------+--------+
4. Eack Option Format

The EACK option is to be used to convey extended acknowledgment information from the receiver to the sender over an established TCP connection.

TCP EACK Option:

Kind: 30

Length: Variable

+--------+--------+
| Kind=30| Length |
|--------+--------+--------+--------+
| Conn n Src Port | Conn n Dst Port |
|--------+--------+--------+--------+
| Conn n Acknowledgement Number |
|--------+--------+--------+--------+
| ... |
|--------+--------+--------+--------+
| Conn n+i Src Port | Conn n+i Dst Port |
|--------+--------+--------+--------+
| Conn n+i Acknowledgement Number |

Conn n Src Port: 16 bits

The source port number of connection n.

Conn n Dst port: 16 bits

The destination port number of connection n.

Conn n Acknowledgement Number: 32 bits

The acknowledgement number of connection n.

Note that the notation of connection n and connection n+i is purely to differentiate different connections. It does not require any numbering scheme for connections.

Each 3-tuple (connection n source port, connection n destination port, connection n acknowledgement number) in EACK option forms a record, referred to as EACK record. Each EACK record describes the
acknowledgement for a connection.

The EACK option is to be sent in a connection by the data receiver to inform the data sender of data of other connection(s) that have been received and queued. The connection that carries the segment with EACK option is referred to as the carrying connection, while connection(s) that is ACKed by the EACK option is(are) referred to as EACKed connections. The carrying connection and ACKed connection(s) MUST belong to the same internet connection group. The EACK option does not change the meaning of the Acknowledgement Number field, and the acknowledgement of the carrying connection (the standard ACK), SHOULD NOT be part of the EACK option.

An EACK option that contains n EACK records will have a length of 8*n+2 bytes, so the 40 bytes available for TCP options can specify a maximum of 4 EACK records.

5. Generating EACK Options: Data Receiver Behavior

If the data receiver has received an EACK-Permitted option in the SYN for a connection, the data receiver MAY elect to generate EACK options on that connection as described below. If the data receiver has not received a EACK-Permitted option for a given connection, it MUST NOT use EACK options on that connection.

If the data receiver elect to generate EACK options, it SHOULD generate an EACK option only when it needs to generate a standard ACK for a connection whether by piggybacking the ACK or generating a separate ACK packet. When the standard ACK is to be generated, the data receiver determines if there are un-ACKed data on other EACK-permitted connections in the internet connection group for that connection. If so, the data receiver may create the option containing up to the maximum number of EACK records allowed by the available option space, each EACK record describing the acknowledgement of such a connection. Each connection whose acknowledgement is sent as an EACK record in EACK option MUST be treated as if it sends the acknowledgement using the standard Acknowledgement Number field without this option.

6. Processing the EACK Option: Data Sender Behavior

When receiving a segment containing an EACK option, the data sender MUST treat each EACK record as if it is a standard ACK received from the connection described in that EACK record. The data sender MUST NOT applies any control bits in this segment to any connections described in the EACK option.
7. EACK Option Examples

The following example attempts to demonstrate the proper behavior of EACK generation by the data receiver.

The case assumes there are 10 connections C1, C2, ..., C10 in an internet connection group. TCP received EACK-Permitted option on all connections except C10. All connections except C9 have unacknowledged received data.

When TCP decides to generate an ACK on C1 connection, it polls the states of other connections in the internet connection group C1 belongs to. TCP finds C2 is a candidate and creates an EACK record for C2. TCP continues the same process and finally creates 4 EACK records for C2, C3, C4 and C5, the maximum allowed by the EACK option. TCP sends the segment with this EACK option and updates the associated states. Then TCP decides to generate an ACK on C6. TCP finds out at this moment only C7, C8 in that internet connection group are EACK-permitted and have unacknowledged data. TCP creates two EACK records for C7 and C8 in the EACK option, sends segment with this EACK option and updates the associated states.

8. IPsec Consideration

IPsec allows the user to control the granularity at which a security service is offered [Ken98]. If the granularity is at a per TCP connection level, EACK may compromise the security of the acknowledgement number of an EACKed connection. Therefore, EACK SHOULD NOT be used on connections on which different security associations apply.

9. IP Network Address Port Translation (NAT) Consideration

A NAT [Sri99] device may translate different IP addresses into a unique IP address. With this translation, an internet connection group, in the perception of the data sender or the data receiver, no longer represents all TCP connections between the data sender and the data receiver. This mismatch causes EACK to fail at least for one direction.

There are two solutions to resolve this issue in the NAT device. One is to block the delivery of EACK-Permitted option; the other is to implement an EACK proxy.

Blocking the delivery of EACK-Permitted option requires the NAT device to remove the EACK-Permitted option from a TCP SYN packet.
Implementing an EACK proxy in a NET device has a much higher complexity. Upon receiving and finishing the address translation for an EACK option, the proxy removes each EACK record for a connection that has a different destination IP address than the carrying connection does and generates a separate ACK packet for the removed connection. The removal of EACK records result in either a shorter EACK option or the removal of the entire EACK option.

In either solution, the implementation MAY change the TCP segment with a shortened TCP header or applies the No-Operation options to keep the length of the TCP header without the need to shift the TCP payload.

10. Authors’ Addresses

Manqing Huang  
Exar Corporation  
48720 Kato Road  
Fremont, CA 94538  
manqing.huang@exar.com

REFERENCES


[All99] Allman, M., Paxson, V. and Stevens, W., "TCP Congestion Control", RFC 2581, April 1999

