TCPM Working Group
Internet-Draft
Intended status: Experimental
Expires: 15 April 2023

C. Gomez
UPC
J. Crowcroft
University of Cambridge

October 2022

TCP ACK Rate Request Option
draft-gomez-tcpm-ack-rate-request-06

Abstract

TCP Delayed Acknowledgments (ACKs) is a widely deployed mechanism that allows reducing protocol overhead in many scenarios. However, Delayed ACKs may also contribute to suboptimal performance. When a relatively large congestion window (cwnd) can be used, less frequent ACKs may be desirable. On the other hand, in relatively small cwnd scenarios, eliciting an immediate ACK may avoid unnecessary delays that may be incurred by the Delayed ACKs mechanism. This document specifies the TCP ACK Rate Request (TARR) option. This option allows a sender to request the ACK rate to be used by a receiver, and it also allows to request immediate ACKs from a receiver.

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 4 April 2023.

Copyright Notice

Table of Contents

1. Introduction

Delayed Acknowledgments (ACKs) were specified for TCP with the aim to reduce protocol overhead [RFC1122]. With Delayed ACKs, a TCP delays sending an ACK by up to 500 ms (often 200 ms, with lower values in recent implementations such as ~50 ms also reported), and typically sends an ACK for at least every second segment received in a stream of full-sized segments. This allows combining several segments into a single one (e.g. the application layer response to an application layer data message, and the corresponding ACK), and also saves up to one of every two ACKs, under many traffic patterns (e.g. bulk transfers). The "SHOULD" requirement level for implementing Delayed ACKs in RFC 1122, along with its expected benefits, has led to a widespread deployment of this mechanism.

However, there exist scenarios where Delayed ACKs contribute to suboptimal performance. We next roughly classify such scenarios into two main categories, in terms of the congestion window (cwnd) size and the Maximum Segment Size (MSS) that would be used therein: i) "large" cwnd scenarios (i.e. cwnd >> MSS), and ii) "small" cwnd scenarios (e.g. cwnd up to ~MSS).

In "large" cwnd scenarios, increasing the number of data segments after which a receiver transmits an ACK beyond the typical one (i.e. 2 when Delayed ACKs are used) may provide significant benefits. One example is mitigating performance limitations due to asymmetric path capacity (e.g. when the reverse path is significantly limited in comparison to the forward path) [RFC3449]. Another advantage is reducing the computational cost both at the sender and the receiver, and reducing network packet load, due to the lower number of ACKs involved.

In many "small" cwnd scenarios, a sender may want to request the receiver to acknowledge a data segment immediately (i.e. without the additional delay incurred by the Delayed ACKs mechanism). In high bit rate environments (e.g. data centers), a flow's fare share of the available Bandwidth Delay Product (BDP) may be in the order of one MSS, or even less. For an accordingly set cwnd value (e.g. cwnd up to MSS), Delayed ACKs would incur a delay that is several orders of magnitude greater than the RTT, severely degrading performance. Note that the Nagle algorithm may produce the same effect for some traffic patterns in the same type of environments [RFC8490]. In addition, when transactional data exchanges are performed over TCP, or when the cwnd size has been reduced, eliciting an immediate ACK from the receiver may avoid idle times and allow timely continuation of data transmission and/or cwnd growth, contributing to maintaining low latency.

Further "small" cwnd scenarios can be found in Internet of Things (IoT) environments. Many IoT devices exhibit significant memory constraints, such as only enough RAM for a send buffer size of 1 MSS. In that case, if the data segment does not elicit an application-layer response, the Delayed ACKs mechanism unnecessarily contributes a delay equal to the Delayed ACK timer to ACK transmission. The sender cannot transmit a new data segment until the ACK corresponding to the previous data segment is received and processed.

With the aim to provide a tool for performance improvement in both "large" and "small" cwnd scenarios, this document specifies the TCP ACK Rate request (TARR) option. This option allows a sender to request the ACK rate to be used by a receiver, and it also allows to request immediate ACKs from a receiver.

2. Conventions used in this document

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. TCP ACK Rate Request Functionality

A TCP endpoint SHOULD announce that it supports the TARR option by including the TARR option format (with the appropriate Length value, see Section 4) in packets that have the SYN bit set.

In some cases (e.g. when SYN cookies are used [RFC4987]), the client MAY announce that it supports the TARR option in packets subsequent to the SYN packet. (Note that announcing TARR option support on the ACK in the three-way handshake is not reliable.)

Upon reception of a segment announcing support of the TARR option, a TARR-option-capable endpoint MUST announce support of the TARR option as well by including it in the next segment to be sent.

The next two subsections define the sender and receiver behaviors for devices that support the TARR option, respectively.

3.1. Sender behavior

A TCP sender MUST NOT include the TARR option in TCP segments to be sent if the TCP receiver does not support the TARR option.

A TCP sender MAY request a TARR-option-capable receiver to modify the ACK rate of the latter to one ACK every R data segments received from the sender. This request is performed by the sender by including the TARR option in the TCP header of a segment. The TARR option carries the R value requested by the sender (see section 4).

When a TCP sender needs a data segment to be acknowledged immediately by a TARR-option-capable receiving TCP, without modifying the steady state ACK rate being used by the receiver, the sender includes the TARR option in the TCP header of the data segment, with a value of R equal to 0.

A TCP segment carrying retransmitted data is not required to include a TARR option.

3.2. Receiver behavior

A receiving TCP conforming to this specification MUST process a TARR option present in a received segment.

A TARR-option-capable receiving TCP SHOULD modify its ACK rate to one ACK every R received data segments from the sender. If a TARR-option-capable TCP receives a segment carrying the TARR option with R=0, the receiving TCP SHOULD send an ACK immediately while keeping its steady state ACK rate.

If packet reordering occurs, a TARR-option-capable receiver should send a duplicate ACK immediately when an out-of-order segment arrives [RFC5681]. After sending a duplicate ACK, the receiver MAY send the next non-duplicate ACK after R data segments received. Note also that the receiver might be unable to send ACKs at the requested rate (e.g., due to lack of resources); on the other hand, the receiver might opt not to fulfill a request for security reasons (e.g., to avoid or mitigate an attack by which a large number of senders request disabling delayed ACKs simultaneously and send a large number of data segments to the receiver).

The request to modify the ACK rate of the receiver holds until the next segment carrying a TARR option is received.

4. Option Format

The TARR option presents two different formats that can be identified by the corresponding format length. For packets that announce TARR option support by their senders, the TARR option has the format shown in Fig. 1.

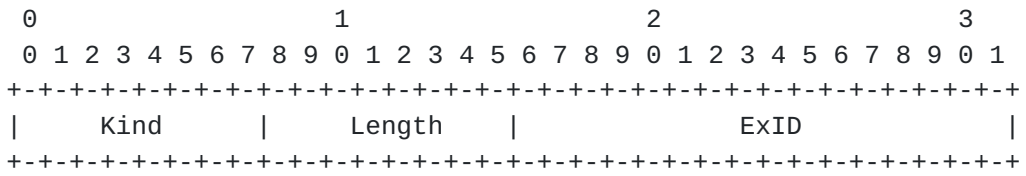

Figure 1: Format used to announce support of the TARR option by the sender.

Kind: The Kind field value is 254.

Length: The Length field value is 4 bytes.

ExID: The experiment ID field size is 2 bytes, and its value is 0x00AC.

For packets that do not have the SYN bit set, the TARR option has the format and content shown in Fig. 2.

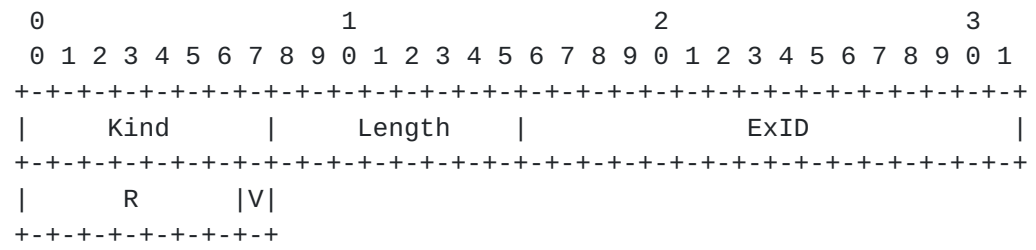

Figure 2: TCP ACK Rate Request option format.

Kind: The Kind field value is 254.

Length: The Length field value is 5 bytes.

ExID: The experiment ID field size is 2 bytes, and its value is 0x00AC.

R: The size of this field is 7 bits. The field carries the binary encoding of the ACK rate requested by the sender. The maximum value of R is 127.

V (reserVed): The size of this field is 1 bit. This field is reserved for future use.

5. Changing the ACK rate during the lifetime of a TCP connection

In some scenarios, setting the ACK rate once for the whole lifetime of a TCP connection may be suitable. However, there are also cases where it may be desirable to modify the ACK rate during the lifetime of a connection.

The ACK rate to be used may depend on the cwnd value used by the sender, which can change over the lifetime of a connection. cwnd will start at a low value and grow rapidly during the slow-start phase, then settle into a reasonably consistent range for the congestion-avoidance phase - assuming the underlying bandwidth-delay product (BDP) remains constant. Phenomena such as routing updates, link capacity changes or path load changes may modify the underlying BDP significantly; the cwnd should be expected to change accordingly, prompting the need for ACK rate updates.

TARR can also be used to suppress Delayed ACKs in order to allow measuring the RTT of each packet in specific intervals (e.g., during flow start-up), and allow a different ACK rate afterwards.

A Linux receiver has a heuristic to detect slow start and suppress Delayed ACKs just for that period. However, some slow start variants (e.g., HyStart, HyStart++, etc.) may alter the ending of slow start, thus confusing the heuristics of the receiver [I-D.ietf-tcpm-hystartplusplus]. To avoid slow start sender behavior ossification, an explicit signal such as TARR may be useful.

Another reason to modify the ACK rate might be reducing the ACK load. The sender may notice that the ACKs it receives cover more segments than the ACK rate requested, indicating that ACK decimation is occurring en route. The sender may then decide to reduce the ACK frequency to reduce receiver workload and network load up to the ACK decimation point.

Future TCP specifications may also permit Congestion Experienced (CE) marks to appear on pure ACKs [I-D.ietf-tcpm-generalized-ecn]. This might involve more frequent ACK rate updates (e.g., once an RTT), as the sender probes around an operating point.

6. IANA Considerations

This document specifies a new TCP option (TCP ACK Rate Request) that uses the shared experimental options format [RFC6994], with ExID in network-standard byte order.

The authors plan to request the allocation of ExID value 0x00AC for the TCP option specified in this document.

7. Security Considerations

The TARR option opens the door to new security threats. This section discusses such new threats, and suggests mitigation techniques.

An attacker might be able to impersonate a legitimate sender, and forge an apparently valid packet intended for the receiver. In such case, the attacker may mount a variety of harmful actions. By using TARR, the attacker may intentionally communicate a bad R value to the latter with the aim to damage communication or device performance. For example, in a small cwnd scenario, using a too high R value may lead to exacerbated RTT increase and throughput decrease. In other scenarios, a too low R value may contribute to depleting the energy of a battery-operated receiver at a faster rate or may lead to increased network packet load.

While Transport Layer Security (TLS) [RFC8446] is strongly recommended for securing TCP-based communication, TLS does not protect TCP headers, and thus cannot protect the TARR option fields carried by a segment. One approach to address the problem is using network-layer protection, such as Internet Protocol Security (IPsec) [RFC4301]. Another solution is using the TCP Authentication Option (TCP-AO), which provides TCP segment integrity and protection against replay attacks [RFC5925].

While it is relatively hard for an off-path attacker to attack an unprotected TCP session, it is RECOMMENDED for a TARR receiver to use the guidance and attack mitigation given in [RFC5961]. The TARR option MUST be ignored on a packet that is deemed invalid.

A TARR receiver might opt not to fulfill a request to avoid or mitigate an attack by which a large number of senders request disabling delayed ACKs simultaneously and send a large number of data segments to the receiver (see Section 3.2).

8. Acknowledgments

Bob Briscoe, Jonathan Morton, Richard Scheffenegger, Neal Cardwell, Michael Tuexen, Yuchung Cheng, Matt Mathis, Jana Iyengar, Gorry Fairhurst, Stuart Cheshire, Yoshifumi Nishida, Michael Scharf, Ian Swett, and Martin Duke provided useful comments and input for this document. Jonathan Morton and Bob Briscoe provided the main input for Section 5.

Carles Gomez has been funded in part by the Spanish Government through project PID2019-106808RA-I00, and by Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement de la Generalitat de Catalunya 2017 through grant SGR 376.

9. References

9.1. Normative 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>.

[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>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <https://www.rfc-editor.org/info/rfc4987>.

[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>.

[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>.

[RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, DOI 10.17487/RFC5961, August 2010, <https://www.rfc-editor.org/info/rfc5961>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, DOI 10.17487/RFC6994, August 2013, <https://www.rfc-editor.org/info/rfc6994>.

[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>.

9.2. Informative References

[I-D.ietf-tcpm-generalized-ecn] Bagnulo, M. and B. Briscoe, "ECN++: Adding Explicit Congestion Notification (ECN) to TCP Control Packets", Work in Progress, Internet-Draft, draft-ietf-tcpm-generalized-ecn-10, 27 July 2022, <https://www.ietf.org/archive/id/draft-ietf-tcpm-generalized-ecn-10.txt>.

[I-D.ietf-tcpm-hystartplusplus] Balasubramanian, P., Huang, Y., and M. Olson, "HyStart++: Modified Slow Start for TCP", Work in Progress, Internet-

Draft, draft-ietf-tcpm-hystartplusplus-10, 3 October 2022, <https://www.ietf.org/archive/id/draft-ietf-tcpm-hystartplusplus-10.txt>.

[RFC3449] Balakrishnan, H., Padmanabhan, V., Fairhurst, G., and M. Sooriyabandara, "TCP Performance Implications of Network Path Asymmetry", BCP 69, RFC 3449, DOI 10.17487/RFC3449, December 2002, <https://www.rfc-editor.org/info/rfc3449>.

[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>.

[RFC5690] Floyd, S., Arcia, A., Ros, D., and J. Iyengar, "Adding Acknowledgement Congestion Control to TCP", RFC 5690, DOI 10.17487/RFC5690, February 2010, <https://www.rfc-editor.org/info/rfc5690>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", RFC 8490, DOI 10.17487/RFC8490, March 2019, <https://www.rfc-editor.org/info/rfc8490>.

Appendix A. Relation between the present document and RFC 5690

A previously published document, entitled "Adding Acknowledgment Congestion Control to TCP" [RFC5690], includes functionality similar to some aspects of the present document. However, the motivation, main goals, and use cases of both documents are almost orthogonal. In fact, some features of the present document were not considered in [RFC5690]. This section compares the main features of RFC 5690 and the present document.

A.1. Motivation, goals and features

RFC 5690 is an informational document that describes a possible congestion control mechanism for TCP ACKs. The main goal is to reduce ACK traffic when there is congestion on the reverse path in order to reduce the congestion. The mechanism includes: i) a component for the TCP sender to detect lost and ECN-marked pure ACKs, ii) a mechanism for adjusting the ACK Ratio, iii) a method to discover the support of the ACK congestion control mechanism by an endpoint (by means of a new TCP option), and iv) a method for the TCP sender to inform the TCP receiver of a new value for the ACK Ratio

(by means of a second new TCP option). As of the writing, and to the best knowledge of the authors, RFC 5690 has not been implemented. Option Kind values for the new TCP options described in RFC 5690 have neither been allocated by IANA.

The present document defines the TARR option. While it can be used to reduce network load, its primary focus is rather on end-to-end performance and end-system resource conservation. TARR serves two purposes: i) allowing a sender to request a given ACK ratio from the receiver, and ii) allowing a sender to request an immediate ACK, without modifying the steady state ACK ratio. The latter is not supported by RFC 5690. On the other hand, TARR might be used as a component of other mechanisms (e.g. an ACK congestion control mechanism). However, such mechanisms are out of the scope of the present document.

A.2. New TCP option details

As part of the ACK congestion control mechanism, RFC 5690 proposes the use of two new TCP options: one intented to announce support of TCP ACK Congestion Control, and another one which is used by the sender to communicate the ACK ratio to the receiver. The former can only be sent on packets that have the SYN bit set. In the latter, a one-byte field is used to carry the ACK ratio, but the encoding to be used for this field is not defined.

The present document uses a single TCP experimental option Kind value (following RFC 6994) for both announcing support of the TARR option, and for communicating the requested ACK ratio. In the present document, announcing support of the TARR option may be done in packets that do not have the SYN bit set, with the aim to alleviate the need for TCP option space in SYN packets. In contrast with RFC 5690, the encoding to be used for the ACK ratio field is specified (see Section 4).

Authors' Addresses

Carles Gomez
UPC
C/Esteve Terradas, 7
08860 Castelldefels
Spain
Email: carles.gomez@upc.edu

Jon Crowcroft
University of Cambridge
JJ Thomson Avenue
Cambridge
United Kingdom
Email: jon.crowcroft@cl.cam.ac.uk