

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: June 20, 2018

W. Eddy
MTI Systems
L. Wood
Surrey alumni
W. Ivancic
Syzygy
December 17, 2017

TFRC-based Congestion Control for Saratoga
draft-eddy-tsvwg-saratoga-tfrc-12

Abstract

This document specifies the use of TCP-Friendly Rate Control (TFRC) with the Saratoga data transfer protocol. The necessary conventions that a Saratoga sender and receiver implementation must follow if they wish to enable the use of TFRC are described.

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). 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 June 20, 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.

Internet-Draft

Saratoga TFRC

December 2017

This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	2
2.	Receiver-Side Behavior	3
3.	Sender-Side Behavior	4
4.	Security Considerations	6
5.	IANA Considerations	6
6.	Acknowledgements	7
7.	References	7
7.1.	Normative References	7
7.2.	Informative References	7
Appendix A.	Differences from Shahriar et al.	7
	Authors' Addresses	8

[1.](#) Introduction

In order to support use of the Saratoga protocol [[draft-wood-tsvwg-saratoga](#)] on networks with multiple data flows, multiple hops, and/or experimentally on the Internet, some form of congestion control is required. Particularly, for use on the Internet, rather than the private links and networks that Saratoga was originally developed for, congestion control is a requirement, as indicated by the UDP Guidelines [[RFC5405](#)].

This document provides the specification of one type of congestion control, based on TCP-Friendly Rate Control (TFRC), which is intended to be fairly conservative in terms of performance. Better-performing congestion-control algorithms are possible in terms of throughput, but this TFRC-based algorithm has the advantages of being relatively simple and based on the well-established TFRC components, rather than on a newer and less well-tested mechanism.

Saratoga data flow occurs within uniquely identified transactions between a sender and receiver. Saratoga primitively supports link-local multicast to a set of receivers, but that use case is not considered further in this specification and is considered out of scope for this document. The congestion control mechanism specified in this document operates within a single transaction between a sender and a receiver. Subsequent or concurrent transactions between

the same nodes use distinct congestion control state.

TFRC has "receiver-based" and "sender-based" variations, as described in [[RFC5348](#)]. In either case, a node with knowledge of the loss-event rate and round-trip time (RTT) uses this information

in the TFRC throughput equation in order to compute an allowed transmit rate for the sender. A sender-based variant of TFRC for use with Saratoga has been described in the academic literature [[Shahriar11](#)], and the mechanism described in this document is derived to some extent from that previous work.

Saratoga allows the receiver to generate, at any time, a STATUS packet containing a "Progress Indicator" cumulative acknowledgement (PI cumack) and a list of "holes" in the sequence space of data bytes received. The STATUS packet can also be requested to be sent by setting a flag in the DATA packet header. A DATA packet can optionally include a timestamp field, which is echoed in any STATUS packet that it may trigger. This enables the sender to estimate the Round-Trip Time (RTT), required as part of TFRC equation. The TFRC adaptation for Saratoga relies on the regular reception of STATUS packets at the sender, in order for the sender to use its knowledge of the RTT, the packets sent within an interval, and the packets received (implicit from the holes and PI cumack), to run the TFRC equation to determine a reasonable sending rate.

Generally, Saratoga is designed to enable high-performance transfers over highly asymmetric and lossy links, which may also have high latencies. The rate-control, in the base specification, has been permitted to be open-loop with the sender configured to saturate the network capacity that it expects exclusive access to. However, in order to permit sender-based TFRC congestion control for shared capacity, accurate and timely feedback is necessary in order to create a functioning closed-loop control system. This limits the applicability of TFRC extensions to Saratoga to networks with RTTs more typical of the Internet than, for instance, interplanetary microwave communications links. Supporting congestion control also limits the asymmetry that can be supported between the data path capacity and the feedback path capacity, as more regular STATUS updates are required for the congestion control loop.

[2.](#) Receiver-Side Behavior

The receiver MUST echo timestamps in non-voluntary (solicited) STATUS packets. It SHOULD generate and transmit these STATUS packets without unnecessary implementation delay. The sender will typically request mandatory STATUS packets at least once per its estimated RTT. If multiple STATUS packets within the same transaction are queued for transmission within the receiver, it MAY choose to only send the most recent STATUS and clear others from its queue. This may help to mitigate asymmetry issues.

The Saratoga receiver MUST track the time when it last sent a STATUS packet, and periodically send voluntary (unsolicited) STATUS packets

to the sender, even if no DATA packets requesting STATUS have been received. This is needed to provide feedback of progress (or lack of progress) when there is heavy loss in the DATA path, and the sender needs to be informed of this in order to adjust its sending rate. This timer for sending unsolicited STATUS packets MAY be set to the minimum among the the last three measured interarrival times between consecutive DATA packets bearing STATUS requests within the transaction.

The receiver SHOULD generate voluntary STATUS packets with an updated holes list when it receives an incoming DATA packets with offset beyond the current cumack (i.e. not advancing the Progress Indicator). This creates a hole in the STATUS packet and provides a timely notification of loss to the sender. Possible exceptions to this behavior would be for highly-asymmetric links, where the feedback traffic needs to be minimized, or for cases where significant reordering is expected and a more intelligent strategy for generating STATUS packets is implemented.

[3.](#) Sender-Side Behavior

As the envisioned use of Saratoga implementations with TFRC is for bulk-transfer applications that will not be "data-limited" in [\[RFC5348\]](#) terms, tracking of data-limited periods and adjusting the sending rate based on them is not required by the specification in this document.

The "nofeedback" timer defined in [\[RFC5348\]](#) is reset based on the reception of Saratoga STATUS packets, as these provide the feedback

information.

It is assumed that the Saratoga sender MUST implement a configurable "Maximum Payload Size" (MPS), which limits the total number of bytes of user data per Saratoga packet. This assumption is used to simplify tracking of lost packets based on STATUS feedback that only provides ranges of DATA holes, and not the detail of individual lost or received packets. The MPS MAY be adjusted downwards by an implementation based on PMTUD feedback, but the details for this are not within scope of this document, and do not impact the TFRC mechanism defined here, as long as the current MPS for a transaction is known by the implementation.

The initial sending rate for DATA packets within a Saratoga transaction from the Saratoga sender implementing TFRC is initialized to:

$$X = \min(4 \cdot \text{MPS}, \max(2 \cdot \text{MPS}, 4380))$$

This is as already defined for TFRC, with the Saratoga transaction MPS used as the Maximum Segment Size that TFRC is based on. During the course of the transaction, this is adjusted as described in the remainder of this section.

The Saratoga sender implementing TFRC-based congestion control specified in this document MUST use timestamps on its DATA packets. For each DATA packet that the Saratoga sender releases onto the network, it MUST temporarily store the timestamp, offset, and packet size in a packet history list. The history is accessed by looking for packets sent at or before a given timestamp, and if organized this way by timestamp, continuous portions of the packet history list are periodically cleared during processing of STATUS packets, described later in this section.

The sender MUST keep track of a per-transaction estimated RTT based on observance of timestamp echo fields in STATUS packets. On receiving a STATUS packet, an RTT sample is computed via subtracting the echoed timestamp from the current clock time. [\[RFC5348\]](#) nominally includes `t_delay`; no `t_delay` is employed (or `t_delay` is 0) as Saratoga does not have a way to convey `t_delay`. The collection of RTT samples SHOULD be smoothed via an algorithm such as the EWMA

described in [[RFC6298](#)]. In this document, the use of "RTT" generally refers to the smoothed RTT computed via this process rather than an instantaneous RTT sample, unless clearly noted.

When sending DATA packets, the sender MUST periodically request STATUS packets using the available DATA packet flag for this purpose. The time period for making these requests is a minimum of once per the current estimated RTT.

There is no assumption of the ordering which a Saratoga sender delivers particular chunks of a file, nor to the order and algorithms which it uses to respond to and repair the list of holes conveyed in the STATUS packets. However, the sender must keep track of the comprehensive set of holes within the transaction that have been seen most recently in STATUS feedback.

On receiving an STATUS packet, the Saratoga sender MUST compute a loss rate sample. The loss rate in packets is computed via comparing the changes between the current snapshot and the prior one generated when the last STATUS packet was received. Signs of positive progress include: (1) advancement of the cumack field by some multiple of the MPS (or less than one MPS in the case of the final packet of DATA within a transaction, (2) reduction in the size of any holes. The number of MPS-sized chunks, `packets_rcvd`, covered by signs of positive progress in the current STATUS packet represents the number of non-duplicate packets received during the last reporting interval

(assuming a prior STATUS packet was not lost). The sender uses its history of sent DATA packets and the timestamp echoed in the STATUS in order to determine how many DATA packets were sent within the interval. It simply counts the number of packets, `packets_sent`, in the history with timestamps prior or equal to the echoed one. At this point, the packet history at or below the echoed timestamp can be cleared or released. `packets_ecnd` is always 0, as implementing support for ECN from a UDP application raises implementation difficulties. The loss event rate sample for the TFRC equation, p , is then computed from `packets_sent`, `packets_rcvd`, and `packets_ecnd` as described below. This loss rate sample, along with the current RTT measurement are fed into the TFRC equation to arrive at a sending rate for the Saratoga sender.

The values needed to compute the TCP sending rate (X_{Bps}) from the

TFRC equation, using the variable names from [\[RFC5348\]](#) are:

- o s := the MPS value configured for the transaction
- o R := the current smoothed RTT
- o p := the computed loss event rate; $1 - ((\text{packets_rcvd} - \text{packets_ecnd}) / \text{packets_sent})$
- o t_RT0 := $4 * R$ (as recommended by [\[RFC5348\]](#))
- o b := 1 (as recommended by [\[RFC5348\]](#))

The TFRC procedure for computing X_recv_set , $recv_limit$, X_Bps , and X are performed as defined in [\[RFC5348\]](#) per STATUS packet received, along with the rest of the TFRC feedback processing. The new X value is immediately applied to the Saratoga implementation's outgoing packet clocking.

[4.](#) Security Considerations

The security considerations for use of TFRC in Saratoga are the same as those described in [\[RFC5348\]](#) for TFRC itself (but do not share the considerations listed there for TFRC's use in DCCP).

[5.](#) IANA Considerations

This document has no IANA considerations.

[6.](#) Acknowledgements

We thank Cisco Systems for funding the early investigations into Saratoga congestion control that led to [\[Shahriar11\]](#), which has heavily influenced this document.

[7.](#) References

7.1. Normative References

[[draft-wood-tsvwg-saratoga](#)]

Wood, L., Eddy, W., Smith, C., Ivancic, W., and C. Jackson, "Saratoga: A Scalable Data Transfer Protocol", [draft-wood-tsvwg-saratoga-22](#) (work in progress) , December 2017.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", [RFC 5348](#), DOI 10.17487/RFC5348, September 2008, <<https://www.rfc-editor.org/info/rfc5348>>.

7.2. Informative References

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", [RFC 5405](#), DOI 10.17487/RFC5405, November 2008, <<https://www.rfc-editor.org/info/rfc5405>>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", [RFC 6298](#), DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.

[Shahriar11]

Shahriar, A., Atiquzzaman, M., Ivancic, W., and L. Wood, "A sender-based TFRC for Saratoga: A rate control mechanism for a space-friendly transfer protocol", IEEE Aerospace Conference Big Sky, Montana, March 2011.

Appendix A. Differences from Shahriar et al.

The method described in this document does not involve keeping track of the "symmetry factor" that was a part of [[Shahriar11](#)].

The method described in this document does not use the "dummy sequences" described as part of [[Shahriar11](#)], and instead infers lost packet counts from the number of bytes covered by holes above the PI cumack (the Progress Indicator). This inference is possible whenever

the cumack is less than the highest sequence number expected (the In-

Response-To field) within the last loss period.

Authors' Addresses

Wesley M. Eddy
MTI Systems

Email: wes@mti-systems.com

Lloyd Wood
University of Surrey alumni
Sydney, New South Wales
Australia

Email: lloydwood@users.sourceforge.net

Will Ivancic
Syzygy Engineering LLC
Westlake, OH 44145
USA

Phone: +1-440-835-8448

Email: ivancic@syzygyengineering.com