Stream: Internet Engineering Task Force (IETF)
RFC: 8836
Category: Informational
Published: September 2020
ISSN: 2070-1721
Authors: R. Jesup, *Mozilla*; Z. Sarker, Ed., *Ericsson*

RFC 8836
Congestion Control Requirements for Interactive Real-Time Media

Abstract

Congestion control is needed for all data transported across the Internet, in order to promote fair usage and prevent congestion collapse. The requirements for interactive, point-to-point real-time multimedia, which needs low-delay, semi-reliable data delivery, are different from the requirements for bulk transfer like FTP or bursty transfers like web pages. Due to an increasing amount of RTP-based real-time media traffic on the Internet (e.g., with the introduction of the Web Real-Time Communication (WebRTC)), it is especially important to ensure that this kind of traffic is congestion controlled.

This document describes a set of requirements that can be used to evaluate other congestion control mechanisms in order to figure out their fitness for this purpose, and in particular to provide a set of possible requirements for a real-time media congestion avoidance technique.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8836.

Copyright Notice

Table of Contents

1. Introduction

Most of today's TCP congestion control schemes were developed with a focus on a use of the Internet for reliable bulk transfer of non-time-critical data, such as transfer of large files. They have also been used successfully to govern the reliable transfer of smaller chunks of data in as short a time as possible, such as when fetching web pages.

These algorithms have also been used for transfer of media streams that are viewed in a non-interactive manner, such as "streaming" video, where having the data ready when the viewer wants it is important, but the exact timing of the delivery is not.

When handling real-time interactive media, the requirements are different. One needs to provide the data continuously, within a very limited time window (no more delay than hundreds of milliseconds end-to-end). In addition, the sources of data may be able to adapt the amount of data that needs sending within fairly wide margins, but they can be rate limited by the application -- even not always having data to send. They may tolerate some amount of packet loss, but since the data is generated in real time, sending "future" data is impossible, and since it's consumed in real time, data delivered late is commonly useless.

While the requirements for real-time interactive media differ from the requirements for the other flow types, these other flow types will be present in the network. The congestion control algorithm for real-time interactive media must work properly when these other flow types are present as cross traffic on the network.

One particular protocol portfolio being developed for this use case is WebRTC [RFC8825], where one envisions sending multiple flows using the Real-time Transport Protocol (RTP) [RFC3550] between two peers, in conjunction with data flows, all at the same time, without having special arrangements with the intervening service providers. As RTP does not provide any congestion control mechanism, a set of circuit breakers, such as those described in [RFC8083], are required to protect the network from excessive congestion caused by non-congestion-controlled flows. When the real-time interactive media is congestion controlled, it is recommended that the congestion control mechanism operate within the constraints defined by these circuit breakers when a circuit breaker is present and that it should not cause congestion collapse when a circuit breaker is not implemented.

Given that this use case is the focus of this document, use cases involving non-interactive media such as video streaming and those using multicast/broadcast-type technologies, are out of scope.

The terminology defined in [RFC8825] is used in this memo.

2. Requirements

1. The congestion control algorithm must attempt to provide as-low-as-possible-delay transit for interactive real-time traffic while still providing a useful amount of bandwidth. There may be lower limits on the amount of bandwidth that is useful, but this is largely application specific, and the application may be able to modify or remove flows in order to allow some useful flows to get enough bandwidth. For example, although there might not be enough bandwidth for low-latency video+audio, there could be enough for audio only.

    a. Jitter (variation in the bitrate over short timescales) is also relevant, though moderate amounts of jitter will be absorbed by jitter buffers. Transit delay should be considered to track the short-term maximums of delay, including jitter.

    b. The algorithm should provide this as-low-as-possible-delay transit and minimize self-induced latency even when faced with intermediate bottlenecks and competing flows. Competing flows may limit what's possible to achieve.

    c. The algorithm should be resilient to the effects of events, such as routing changes, which may alter or remove bottlenecks or change the bandwidth available, especially if there is a reduction in available bandwidth or increase in observed delay. It is expected that the mechanism reacts quickly to such events to avoid delay buildup. In the context of this memo, a "quick" reaction is on the order of a few RTTs, subject to the constraints of the media codec, but is likely within a second. Reaction on the next RTT is explicitly not required, since many codecs cannot adapt their sending rate that quickly, but at the same time a response cannot be arbitrarily delayed.

    d. The algorithm should react quickly to handle both local and remote interface changes (e.g., WLAN to 3G data) that may radically change the bandwidth available or bottlenecks,

especially if there is a reduction in available bandwidth or an increase in bottleneck delay. It is assumed that an interface change can generate a notification to the algorithm.

e. The real-time interactive media applications can be rate limited. This means the offered loads can be less than the available bandwidth at any given moment and may vary dramatically over time, including dropping to no load and then resuming a high load, such as in a mute/unmute operation. Hence, the algorithm must be designed to handle such behavior from a media source or application. Note that the reaction time between a change in the bandwidth available from the algorithm and a change in the offered load is variable, and it may be different when increasing versus decreasing.

f. The algorithm is required to avoid building up queues when competing with short-term bursts of traffic (for example, traffic generated by web browsing), which can quickly saturate a local-bottleneck router or link but clear quickly. The algorithm should also react quickly to regain its previous share of the bandwidth when the local bottleneck or link is cleared.

g. Similarly, periodic bursty flows such as MPEG DASH [MPEG_DASH] or proprietary media streaming algorithms may compete in bursts with the algorithm and may not be adaptive within a burst. They are often layered on top of TCP but use TCP in a bursty manner that can interact poorly with competing flows during the bursts. The algorithm must not increase the already existing delay buildup during those bursts. Note that this competing traffic may be on a shared access link, or the traffic burst may cause a shift in the location of the bottleneck for the duration of the burst.

2. The algorithm must be fair to other flows, both real-time flows (such as other instances of itself) and TCP flows, both long-lived flows and bursts such as the traffic generated by a typical web-browsing session. Note that "fair" is a rather hard-to-define term. It should be fair with itself, giving a fair share of the bandwidth to multiple flows with similar RTTs, and if possible to multiple flows with different RTTs.

   a. Existing flows at a bottleneck must also be fair to new flows to that bottleneck and must allow new flows to ramp up to a useful share of the bottleneck bandwidth as quickly as possible. A useful share will depend on the media types involved, total bandwidth available, and the user-experience requirements of a particular service. Note that relative RTTs may affect the rate at which new flows can ramp up to a reasonable share.

3. The algorithm should not starve competing TCP flows and should, as best as possible, avoid starvation by TCP flows.

   a. The congestion control should prioritize achieving a useful share of the bandwidth depending on the media types and total available bandwidth over achieving as-low-as-possible transit delay, when these two requirements are in conflict.

4. The algorithm should adapt as quickly as possible to initial network conditions at the start of a flow. This should occur whether the initial bandwidth is above or below the bottleneck bandwidth.

    a. The algorithm should allow different modes of adaptation; for example, the startup adaptation may be faster than adaptation later in a flow. It should allow for both slow-start operation (adapt up) and history-based startup (start at a point expected to be at or below channel bandwidth from historical information, which may need to adapt down quickly if the initial guess is wrong). Starting too low and/or adapting up too slowly can cause a critical point in a personal communication to be poor ("Hello!"). Starting too high above the available bandwidth causes other problems for user experience, so there's a tension here. Alternative methods to help startup, such as probing during setup with dummy data, may be useful in some applications; in some cases, there will be a considerable gap in time between flow creation and the initial flow of data. Again, a flow may need to change adaptation rates due to network conditions or changes in the provided flows (such as unmuting or sending data after a gap).

5. The algorithm should be stable if the RTP streams are halted or discontinuous (for example, when using Voice Activity Detection).

    a. After stream resumption, the algorithm should attempt to rapidly regain its previous share of the bandwidth; the aggressiveness with which this is done will decay with the length of the pause.

6. Where possible, the algorithm should merge information across multiple RTP streams sent between two endpoints when those RTP streams share a common bottleneck, whether or not those streams are multiplexed onto the same ports. This will allow congestion control of the set of streams together instead of as multiple independent streams. It will also allow better overall bandwidth management, faster response to changing conditions, and fairer sharing of bandwidth with other network users.

    a. The algorithm should also share information and adaptation with other non-RTP flows between the same endpoints, such as a WebRTC data channel [RFC8831], when possible.
    b. When there are multiple streams across the same 5-tuple coordinating their bandwidth use and congestion control, the algorithm should allow the application to control the relative split of available bandwidth. The most correlated bandwidth usage would be with other flows on the same 5-tuple, but there may be use in coordinating measurement and control of the local link(s). Use of information about previous flows, especially on the same 5-tuple, may be useful input to the algorithm, especially regarding startup performance of a new flow.

7. The algorithm should not require any special support from network elements to be able to convey congestion-related information. As much as possible, it should leverage available information about the incoming flow to provide feedback to the sender. Examples of this information are the packet arrival times, acknowledgements and feedback, packet

timestamps, packet losses, and Explicit Congestion Notification (ECN) [RFC3168]; all of these can provide information about the state of the path and any bottlenecks. However, the use of available information is algorithm dependent.

a. Extra information could be added to the packets to provide more detailed information on actual send times (as opposed to sampling times), but such information should not be required.

8. Since the assumption here is a set of RTP streams, the backchannel typically should be done via the RTP Control Protocol (RTCP) [RFC3550]; instead, one alternative would be to include it in a reverse-RTP channel using header extensions.

   a. In order to react sufficiently quickly when using RTCP for a backchannel, an RTP profile such as RTP/AVPF [RFC4585] or RTP/SAVPF [RFC5124] that allows sufficiently frequent feedback must be used. Note that in some cases, backchannel messages may be delayed until the RTCP channel can be allocated enough bandwidth, even under AVPF rules. This may also imply negotiating a higher maximum percentage for RTCP data or allowing solutions to violate or modify the rules specified for AVPF.
   b. Bandwidth for the feedback messages should be minimized using techniques such as those in [RFC5506], to allow RTCP without Sender/Receiver Reports.
   c. Backchannel data should be minimized to avoid taking too much reverse-channel bandwidth (since this will often be used in a bidirectional set of flows). In areas of stability, backchannel data may be sent more infrequently so long as algorithm stability and fairness are maintained. When the channel is unstable or has not yet reached equilibrium after a change, backchannel feedback may be more frequent and use more reverse-channel bandwidth. This is an area with considerable flexibility of design, and different approaches to backchannel messages and frequency are expected to be evaluated.

9. Flows managed by this algorithm and flows competing against each other at a bottleneck may have different Differentiated Services Code Point (DSCP) [RFC5865] markings depending on the type of traffic or may be subject to flow-based QoS. A particular bottleneck or section of the network path may or may not honor DSCP markings. The algorithm should attempt to leverage DSCP markings when they're available.

10. The algorithm should sense the unexpected lack of backchannel information as a possible indication of a channel-overuse problem and react accordingly to avoid burst events causing a congestion collapse.
11. The algorithm should be stable and maintain low delay when faced with Active Queue Management (AQM) algorithms. Also note that these algorithms may apply across multiple queues in the bottleneck or to a single queue.

3. Deficiencies of Existing Mechanisms

Among the existing congestion control mechanisms, TCP Friendly Rate Control (TFRC) [RFC5348] is the one that claims to be suitable for real-time interactive media. TFRC is an equation-based congestion control mechanism that provides a reasonably fair share of bandwidth when

competing with TCP flows and offers much lower throughput variations than TCP. This is achieved by a slower response to the available bandwidth change than TCP. TFRC is designed to perform best with applications that have a fixed packet size and do not have a fixed period between sending packets.

TFRC detects loss events and reacts to congestion-caused loss by reducing its sending rate. It allows applications to increase the sending rate until loss is observed in the flows. As noted in IAB/IRTF report [RFC7295], large buffers are available in the network elements, which introduce additional delay in the communication. It becomes important to take all possible congestion indications into consideration. Looking at the current Internet deployment, TFRC's biggest deficiency is that it only considers loss events as a congestion indication.

A typical real-time interactive communication includes live-encoded audio and video flow(s). In such a communication scenario, an audio source typically needs a fixed interval between packets and needs to vary the segment size of the packets instead of the packet rate in response to congestion; therefore, it sends smaller packets. A variant of TFRC, Small-Packet TFRC (TFRC-SP) [RFC4828], addresses the issues related to such kind of sources. A video source generally varies video frame sizes, can produce large frames that need to be further fragmented to fit into path Maximum Transmission Unit (MTU) size, and has an almost fixed interval between producing frames under a certain frame rate. TFRC is known to be less optimal when using such video sources.

There are also some mismatches between TFRC's design assumptions and how the media sources in a typical real-time interactive application work. TFRC is designed to maintain a smooth sending rate; however, media sources can change rates in steps for both rate increase and rate decrease. TFRC can operate in two modes: i) bytes per second and ii) packets per second, where typical real-time interactive media sources operate on bit per second. There are also limitations on how quickly the media sources can adapt to specific sending rates. Modern video encoders can operate in a mode in which they can vary the output bitrate a lot depending on the way they are configured, the current scene they are encoding, and more. Therefore, it is possible that the video source will not always output at an allowable bitrate. TFRC tries to increase its sending rate when transmitting at the maximum allowed rate, and it increases only twice the current transmission rate; hence, it may create issues when the video sources vary their bitrates.

Moreover, there are a number of studies on TFRC that show its limitations, including TFRC's unfairness to low statistically multiplexed links, oscillatory behavior, performance issues in highly dynamic loss-rate conditions, and more [CH09].

Looking at all these deficiencies, it can be concluded that the requirements for a congestion control mechanism for real-time interactive media cannot be met by TFRC as defined in the standard.

4. IANA Considerations

This document has no IANA actions.

5. Security Considerations

An attacker with the ability to delete, delay, or insert messages into the flow can fake congestion signals, unless they are passed on a tamper-proof path. Since some possible algorithms depend on the timing of packet arrival, even a traditional, protected channel does not fully mitigate such attacks.

An attack that reduces bandwidth is not necessarily significant, since an on-path attacker could break the connection by discarding all packets. Attacks that increase the perceived available bandwidth are conceivable and need to be evaluated. Such attacks could result in starvation of competing flows and permit amplification attacks.

Algorithm designers should consider the possibility of malicious on-path attackers.

6. References

6.1. Normative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/rfc4585>.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <https://www.rfc-editor.org/info/rfc5124>.

[RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, September 2020, <https://www.rfc-editor.org/info/rfc8825>.

6.2. Informative References

[CH09] Choi, S. and M. Handley, "Designing TCP-Friendly Window-based Congestion Control for Real-time Multimedia Applications", Proceedings of PFLDNeT, May 2009.

[MPEG_DASH] ISO, "Information Technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats", ISO/IEC 23009-1:2019, December 2019, <https://www.iso.org/standard/79329.html>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC4828] Floyd, S. and E. Kohler, "TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant", RFC 4828, DOI 10.17487/RFC4828, April 2007, <https://www.rfc-editor.org/info/rfc4828>.

[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>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <https://www.rfc-editor.org/info/rfc5506>.

[RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, DOI 10.17487/RFC5865, May 2010, <https://www.rfc-editor.org/info/rfc5865>.

[RFC7295] Tschofenig, H., Eggert, L., and Z. Sarker, "Report from the IAB/IRTF Workshop on Congestion Control for Interactive Real-Time Communication", RFC 7295, DOI 10.17487/RFC7295, July 2014, <https://www.rfc-editor.org/info/rfc7295>.

[RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <https://www.rfc-editor.org/info/rfc8083>.

[RFC8831] Jesup, R., Loreto, S., and M. Tüxen, "WebRTC Data Channels", RFC 8831, DOI 10.17487/RFC8831, September 2020, <https://www.rfc-editor.org/info/rfc8831>.

Acknowledgements

This document is the result of discussions in various fora of the WebRTC effort, in particular on the <rtp-congestion@alvestrand.no> mailing list. Many people contributed their thoughts to this.

Authors' Addresses

Randell Jesup
Mozilla
United States of America
Email: randell-ietf@jesup.org

Zaheduzzaman Sarker (EDITOR)
Ericsson
Sweden
Email: zaheduzzaman.sarker@ericsson.com