Network Working Group R. Jesup
Internet-Draft Mozilla
Intended status: Informational March 31, 2014
Expires: October 2, 2014

Congestion Control Requirements For RMCAT
draft-ietf-rmcat-cc-requirements-03

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.

This document attempts to describe 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 proposals coming out of the RMCAT Working Group.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Copyright Notice

Table of Contents

1. Introduction

The traditional TCP congestion control requirements were developed in order to promote efficient 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 doing real time interactive media, the requirements are different; one needs to provide the data continuously, within a very limited time window (no more than 100s of milliseconds end-to-end delay), the sources of data may be able to adapt the amount of data that needs sending within fairly wide margins, and 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 RMCAT differ from the requirements for the other flow types, these other flow types will be present in the network. The RMCAT congestion control algorithm must work properly when these other flow types are present as cross traffic on the network.

One particular protocol portofolio being developed for this use case is WebRTC [I-D.ietf-rtcweb-overview], where one envisions sending multiple RTP-based flows between two peers, in conjunction with data flows, all at the same time, without having special arrangements with the intervening service providers.

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

The terminology defined in [I-D.ietf-rtcweb-overview] is used in this memo.

2. Requirements

1. The congestion control algorithm must attempt to provide as-low-as-possible-delay transit for 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 allow some useful flows to get enough bandwidth. (Example: not enough bandwidth for low-latency video+audio, but enough for audio-only.)

    A. It should provide this as-low-as-possible-delay transit even when faced with intermediate bottlenecks and competing flows. Competing flows may limit what's possible to achieve.

B. It should handle routing changes which may alter or remove bottlenecks or change the bandwidth available, and react quickly, especially if there is a reduction in available bandwidth or increase in bottleneck delay.

C. It should handle interface changes (WiFi to 3G data, etc) which may radically change the bandwidth available or bottlenecks, and react quickly, especially if there is a reduction in available bandwidth or increase in bottleneck

delay. It is assumed that an interface change can generate a notification to the algorithm.

D. The offered load may 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 operation. The reaction time between a change in the bandwidth available from the algorithm and a change in the offered load is variable, and may be different when increasing versus decreasing.

E. The algorithm must not overreact to short-term bursts (such as web-browsing) which can quickly saturate a local-bottleneck router or link, but also clear quickly, and should recover quickly when the burst ends. This is inherently at odds with the need to react quickly-enough to avoid queue buildup.

F. 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 are layered on top of TCP. The algorithm must avoid too much delay buildup during those bursts, and quickly recover. Note that this traffic may on an access link, or 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 realtime flows (such as other instances of itself), and TCP flows, both long-

lived 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 self-fair with itself, giving roughly equal 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 quickly. Note that relative RTTs may affect the rate new flows can ramp up to a reasonable share.

3. The algorithm should where possible merge information across multiple RTP streams between the same endpoints, whether or not they're multiplexed on the same ports, in order to allow congestion control of the set of streams together instead of as multiple independent streams. This allows better overall bandwidth management, faster response to changing conditions, and fairer sharing of bandwidth with other network users.

Alternatively, it should work with an external bandwidth control framework to coordinate bandwidth usage across a bottleneck, such as draft-welzl-rmcat-coupled-cc [I-D.welzl-rmcat-coupled-cc].

A. If possible, it should also share information and adaptation with other non-RTP flows between the same endpoints, such as a WebRTC DataChannel[I-D.ietf-rtcweb-data-channel]

B. 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).

C. Use of information about previous flows, especially on the same 5-tuple, may be useful input to the algorithm, especially to startup performance of a new flow.

D. When there are multiple streams across the same 5-tuple coordinating their bandwidth use and congestion control, it should be possible for the application to control the relative split of available bandwidth.

4. The algorithm should not require any special support from network elements (ECN, etc). 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 ECN, packet arrival times, acknowledgments and feedback, packet timestamps, and packet losses; all of these can provide information about the state of the path and any bottlenecks.

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

    B. When additional input signals such as ECN are available, they should be utilized if possible.

5. Since the assumption here is a set of RTP streams, the backchannel typically should be done via RTCP; one alternative would be to include it instead 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.

    B. 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 RMCAT solutions to violate or modify the rules specified for AVPF.

    C. Bandwidth for the feedback messages should be minimized (such as via RFC 5506 [RFC5506]to allow RTCP without SR/RR)

    D. Header extensions would avoid the RTCP timing rules issues, and allow the application to allocate bandwidth as needed for the congestion algorithm.

    E. 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.

6. Flows managed by this algorithm and flows competing against at a bottleneck may have different 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.

   A. In WebRTC, a division of packets into 4 classes is envisioned in order of priority: faster-than-audio, audio, video, best-effort, and bulk-transfer. Typically the flows managed by this algorithm would be audio or video in that heirarchy, and feedback flows would be faster-than-audio.

7. 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.

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

   A. An algorithm may be more successful at avoiding starvation from short-lived TCP long-lived/saturating TCP flows.

   B. In order to avoid starvation other goals may need to be compromised (such as delay).

9. The algorithm should be stable and 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

10. The algorithm should quickly adapt to initial network conditions at the start of a flow. This should occur both if the initial bandwidth is above or below the bottleneck bandwidth.

    A. 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 over-bandwidth causes other problems for user experience, so there's a tension here.

    B. Alternative methods to help startup like 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.

    C. A flow may need to change adaptation rates due to network conditions or changes in the provided flows (such as un-muting or sending data after a gap).

11. It should be evaluated in how it works both with backbone-router bottlenecks, (asymmetric) local-loop bottlenecks, and local-lan (WiFi/etc) bottlenecks, and in competition with varying numbers and types of streams (TCP, TCP variants in use, LEDBAT [I-D.ietf-ledbat-congestion], inflexible VoIP UDP flows).

12. It should be stable if the RTP streams are halted or discontinuous (VAD/DTX).

    A. After a resumption of RTP data it may adapt more quickly (similar to the start of a flow), and previous bandwidth estimates may need to be aged or thrown away.

3. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

4. Security Considerations

An attacker with the ability to delete, delay or insert messages in 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 percieved available bandwidth are concievable, and need to be evaluated.

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

5. 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.

6. References

6.1. Normative References

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Brower-based Applications", draft-ietf-rtcweb-overview-09 (work in progress), February 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[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, July 2006.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

6.2. Informative References

[I-D.ietf-ledbat-congestion]
Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", draft-ietf-ledbat-congestion-10 (work in progress), September 2012.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-07 (work in progress), February 2014.

[I-D.welzl-rmcat-coupled-cc]
Welzl, M., Islam, S., and S. Gjessing, "Coupled congestion control for RTP media", draft-welzl-rmcat-coupled-cc-02 (work in progress), October 2013.

[MPEG_DASH]
"Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats", April 2012.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

[RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, May 2010.

Author's Address

Randell Jesup
Mozilla
USA

Email: randell-ietf@jesup.org