        Congestion Control Requirements for Interactive Real-Time Media
                  draft-ietf-rmcat-cc-requirements-08

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 real-time media congestion avoidance
   technique.

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].
   The terms are presented in many cases using lowercase for
   readability.

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 May 15, 2015.

Copyright Notice

Table of Contents

1. Introduction

Most of today's TCP congestion control schemes were developed with a
focus on an 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 but can be rate limited
by the application- even not always have data to send, 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 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 [I-D.ietf-rtcweb-overview], 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
[I-D.ietf-avtcore-rtp-circuit-breakers], are required to protect the
network from excessive congestion caused by the non-congestion
controlled flows. When the real-time interactive media is congestion
controlled, it is recommended that the congestion control mechanism
operates within the constraints defined by these circuit breakers
when circuit breaker is present and that it should not cause
congestion collapse when 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 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 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 allow some useful flows
     to get enough bandwidth. (Example: not enough bandwidth for
     low-latency video+audio, but enough for audio-only.)

A. Jitter (variation in the bitrate over short timescales) also
    is 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. It 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. It should handle the effect of 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 the routing change
    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 equally response cannot be arbitrarily
    delayed.

D. It should react quickly to handle both local and remote
    interface changes (WLAN to 3G data, etc) which may radically
    change the bandwidth available or bottlenecks, 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.

E. The algorithm must consider the case where offered loads are
    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. 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 may be different when
    increasing versus decreasing.

F. The algorithm requires 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 also 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 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 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 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 prioritise 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 as quickly as possible 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 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
over-bandwidth causes other problems for user experience, so
there's a tension here. 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. Again, 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).

5.   The algorithm should be stable if the RTP streams are halted or
      discontinuous (for example - 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.   The algorithm should where possible 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, 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.

      A.   The algorithm 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], 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 to startup
           performance of a new flow.

7.   The algorithm should not require any special support from
      network elements to convey congestion related information to be
      functional. 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, and
packet losses, 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 should not be required.

8.   Since the assumption here is a set of RTP streams, the
     backchannel typically should be done via RTCP[RFC3550]; 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. 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
       (such as via RFC 5506 [RFC5506]to allow RTCP without Sender
       Reports/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 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. The algorithm should attempt to
leverage DSCP markings when they're available.

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
hierarchy, and feedback flows would be faster-than-audio.

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 which claims to be suitable for
real-time interactive media. TFRC is, an equation based, congestion
control mechanism which provides reasonably fair share of the
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 which has fixed packet
size and does not have fixed period between sending packets.

TFRC operates on detecting loss events and reacts to loss caused by
congestion by reducing its sending rate. It allows applications to
increase the sending rate until loss is observed in the flows. As it
is noted in IAB/IRTF report [RFC7295] large buffers are available in
the network elements which introduces additional delay in the
communication, it becomes important to take all possible congestion
indications into considerations. Looking at the current Internet
deployment, TFRC's only consideration of loss events as congestion
indication can be considered as biggest lacking.

A typical real-time interactive communication includes live encoded
audio and video flow(s). In such a communication scenario an audio
source typically needs fixed interval between packets, needs to vary
their segment size instead of their packet rate in response to
congestion and 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 which need to be further fragmented to fit
into path Maximum Transmission Unit (MTU) size, and have almost fixed
interval between producing frames under a certain frame rate, TFRC is
known to be less optimal when using with 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
works. TFRC is design to maintain 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 operates on bit per second. There are also limitations on
how quickly the media sources can adapt to specific sending rates.
The modern video encoders can operate on a mode where they can vary
the output bitrate a lot depending on the way there are configured,
the current scene it is encoding and more. Therefore, it is possible
that the video source does not always output at a bitrate they are
allowed to. TFRC tries to raise its sending rate when transmitting
at maximum allowed rate and increases only twice the current
transmission rate hence it may create issues when the video source
vary their bitrates.

Moreover, there are number of studies on TFRC which shows it's
limitations which includes TFRC's unfairness on low statistically
multiplexed links, oscillatory behavior, performance issue in highly
dynamic loss rates conditions and more [CH09].

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

4. IANA Considerations

This document makes no request of IANA.

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

5. 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 perceived available bandwidth
are conceivable, and need to be evaluated.

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

6. 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.

7. References

7.1. Normative References

[I-D.ietf-rtcweb-overview]
          Alvestrand, H., "Overview: Real Time Protocols for
          Browser-based Applications", draft-ietf-rtcweb-overview-12
          (work in progress), October 2014.

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
          Jacobson, "RTP: A Transport Protocol for Real-Time
          Applications", STD 64, RFC 3550, July 2003.

[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.

7.2. Informative References

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

[I-D.ietf-avtcore-rtp-circuit-breakers]
          Perkins, C. and V. Singh, "Multimedia Congestion Control:
          Circuit Breakers for Unicast RTP Sessions", draft-ietf-
          avtcore-rtp-circuit-breakers-07 (work in progress),
          October 2014.

   [I-D.ietf-rtcweb-data-channel]
              Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data
              Channels", draft-ietf-rtcweb-data-channel-12 (work in
              progress), September 2014.

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

   [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition
              of Explicit Congestion Notification (ECN) to IP", RFC
              3168, September 2001.

   [RFC4828] Floyd, S. and E. Kohler, "TCP Friendly Rate Control
              (TFRC): The Small-Packet (SP) Variant", RFC 4828, April
              2007.

   [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP
              Friendly Rate Control (TFRC): Protocol Specification", RFC
              5348, September 2008.

   [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.

   [RFC7295] Tschofenig, H., Eggert, L., and Z. Sarker, "Report from
              the IAB/IRTF Workshop on Congestion Control for
              Interactive Real-Time Communication", RFC 7295, July 2014.

Authors' Addresses

   Randell Jesup
   Mozilla
   USA

   Email: randell-ietf@jesup.org

   Zaheduzzaman Sarker (editor)
   Ericsson
   Sweden

   Email: zaheduzzaman.sarker@ericsson.com