

Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 27, 2012

H. Lundin
S. Holmer
H. Alvestrand, Ed.
Google
April 25, 2012

A Google Congestion Control Algorithm for Real-Time Communication on the
World Wide Web
draft-alvestrand-rtcweb-congestion-02

Abstract

This document describes two methods of congestion control when using real-time communications on the World Wide Web (RTCWEB); one sender-based and one receiver-based.

It is published to aid the discussion on mandatory-to-implement flow control for RTCWEB applications; initial discussion is expected in the RTCWEB WG's mailing list.

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].

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 <http://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 October 27, 2012.

Copyright Notice

Copyright (c) 2012 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 (<http://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. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Mathematical notation conventions	3
2. System model	4
3. Receiver side control	5
3.1. Arrival-time model	5
3.2. Arrival-time filter	6
3.3. Over-use detector	8
3.4. Rate control	8
4. Sender side control	11
5. Interoperability Considerations	12
6. Implementation Experience	13
7. Further Work	13
8. IANA Considerations	14
9. Security Considerations	14
10. Acknowledgements	15
11. References	15
11.1. Normative References	15
11.2. Informative References	15
Appendix A. Change log	16
A.1. Version -00 to -01	16
A.2. Version -01 to -02	16
Authors' Addresses	16

1. Introduction

Congestion control is a requirement for all applications that wish to share the Internet [RFC2914].

The problem of doing congestion control for real-time media is made difficult for a number of reasons:

- o The media is usually encoded in forms that cannot be quickly changed to accommodate varying bandwidth, and bandwidth requirements can often be changed only in discrete, rather large steps
- o The participants may have certain specific wishes on how to respond - which may not be reducing the bandwidth required by the flow on which congestion is discovered
- o The encodings are usually sensitive to packet loss, while the real time requirement precludes the repair of packet loss by retransmission

This memo describes two congestion control algorithms that together are seen to give reasonable performance and reasonable (not perfect) bandwidth sharing with other conferences and with TCP-using applications that share the same links.

The signalling used consists of standard RTP timestamps [RFC3550] possibly augmented with RTP transmission time offsets [RFC5450], standard RTCP feedback reports and Temporary Maximum Media Stream Bit Rate Requests (TMMBR) as defined in [RFC5104] section 3.5.4, or by using the REMB feedback report defined in [I-D.alvestrand-rmcat-remb]

1.1. Mathematical notation conventions

The mathematics of this document have been transcribed from a more formula-friendly format.

The following notational conventions are used:

\bar{X} The variable X , where X is a vector - conventionally marked by a bar on top of the variable name.

\hat{X} An estimate of the true value of variable X - conventionally marked by a circumflex accent on top of the variable name.

$X(i)$ The "i"th value of X - conventionally marked by a subscript i .

$[x\ y\ z]$ A row vector consisting of elements x , y and z .

X_{bar}^T The transpose of vector X_{bar} .

$E\{X\}$ The expected value of the stochastic variable X

2. System model

The following elements are in the system:

- o RTP packet - an RTP packet containing media data.
- o Frame - a set of RTP packets transmitted from the sender at the same time instant. This could be a video frame, an audio frame, or a mix of audio and video packets. A frame can be defined by the RTP packet send time (RTP timestamp + transmission time offset), or by the RTP timestamp if the transmission time offset field is not present.
- o Incoming media streams - a stream of frames consisting of RTP packets.
- o Media codec - has a bandwidth control, and encodes the incoming media stream into an RTP stream.
- o RTP sender - sends the RTP stream over the network to the RTP receiver. Generates the RTP timestamp.
- o RTP receiver - receives the RTP stream, notes the time of arrival. Regenerates the media stream for the recipient.
- o RTCP sender at RTP sender - sends sender reports.
- o RTCP sender at RTP receiver - sends receiver reports and TMMBR/REMB messages.
- o RTCP receiver at RTP sender - receives receiver reports and TMMBR/REMB messages, reports these to sender side control.
- o RTCP receiver at RTP receiver.
- o Sender side control - takes loss rate info, round trip time info, and TMMBR/REMB messages and computes a sending bitrate.

- o Receiver side control - takes the packet arrival info at the RTP receiver and decides when to send TMMBR/REMB messages.

Together, sender side control and receiver side control implement the congestion control algorithm.

3. Receiver side control

The receive-side algorithm can be further decomposed into three parts: an arrival-time filter, an over-use detector, and a remote rate-control.

3.1. Arrival-time model

This section describes an adaptive filter that continuously updates estimates of network parameters based on the timing of the received frames.

At the receiving side we are observing groups of incoming packets, where each group of packets corresponding to the same frame having timestamp $T(i)$.

Each frame is assigned a receive time $t(i)$, which corresponds to the time at which the whole frame has been received (ignoring any packet losses). A frame is delayed relative to its predecessor if $t(i)-t(i-1) > T(i)-T(i-1)$, i.e., if the arrival time difference is larger than the timestamp difference.

We define the (relative) inter-arrival time, $d(i)$ as

$$d(i) = t(i) - t(i-1) - (T(i) - T(i-1))$$

Since the time t_s to send a frame of size L over a path with a capacity of C is roughly

$$t_s = L/C$$

we can model the inter-arrival time as

$$d(i) = \frac{L(i) - L(i-1)}{C} + w(i) = dL(i)/C + w(i)$$

Here, $w(i)$ is a sample from a stochastic process W , which is a

function of the capacity C , the current cross traffic $X(i)$, and the current send bit rate $R(i)$. We model W as a white Gaussian process. If we are over-using the channel we expect $w(i)$ to increase, and if a queue on the network path is being emptied, $w(i)$ will decrease; otherwise the mean of $w(i)$ will be zero.

Breaking out the mean $m(i)$ from $w(i)$ to make the process zero mean, we get

Equation 5

$$d(i) = dL(i)/C + m(i) + v(i)$$

This is our fundamental model, where we take into account that a large frame needs more time to traverse the link than a small frame, thus arriving with higher relative delay. The noise term represents network jitter and other delay effects not captured by the model.

When graphing the values for $d(i)$ versus $dL(i)$ on a scatterplot, we find that most samples cluster around the center, and the outliers are clustered along a line with average slope $1/C$ and zero offset.

For instance, when using a regular video codec, most frames are roughly the same size after encoding (the central "cloud"); the exceptions are I-frames (or key frames) which are typically much larger than the average causing positive outliers (the I-frame itself) and negative outliers (the frame after an I-frame) on the dL axis. Audio frames on the other hand often consist of single packets of equal size, and an audio-only media stream would have its frames scattered at $dL = 0$.

3.2. Arrival-time filter

The parameters $d(i)$ and $dL(i)$ are readily available for each frame $i > 1$, and we want to estimate $C(i)$ and $m(i)$ and use those estimates to detect whether or not we are over-using the bandwidth currently available. These parameters are easily estimated by any adaptive filter - we are using the Kalman filter.

Let

$$\theta_{\text{bar}}(i) = [1/C(i) \quad m(i)]^T$$

and call it the state of time i . We model the state evolution from time i to time $i+1$ as

$$\theta_{\text{bar}}(i+1) = \theta_{\text{bar}}(i) + u_{\text{bar}}(i)$$

where $u_{\text{bar}}(i)$ is the zero mean white Gaussian process noise with covariance

Equation 7

$$Q(i) = E\{u_{\text{bar}}(i) u_{\text{bar}}(i)^T\}$$

Given equation 5 we get

Equation 8

$$d(i) = h_{\text{bar}}(i)^T \theta_{\text{bar}}(i) + v(i)$$

$$h_{\text{bar}}(i) = [dL(i) \quad 1]^T$$

where $v(i)$ is zero mean white Gaussian measurement noise with variance $\text{var}_v = \sigma(v,i)^2$

The Kalman filter recursively updates our estimate

$$\theta_{\text{hat}}(i) = [1/C_{\text{hat}}(i) \quad m_{\text{hat}}(i)]^T$$

as

$$z(i) = d(i) - h_{\text{bar}}(i)^T * \theta_{\text{hat}}(i-1)$$

$$\theta_{\text{hat}}(i) = \theta_{\text{hat}}(i-1) + z(i) * k_{\text{bar}}(i)$$

$$k_{\text{bar}}(i) = \frac{E(i-1) * h_{\text{bar}}(i)}{\text{var}_v_{\text{hat}} + h_{\text{bar}}(i)^T * E(i-1) * h_{\text{bar}}(i)}$$

$$E(i) = (I - K_{\text{bar}}(i) * h_{\text{bar}}(i)^T) * E(i-1) + Q(i)$$

I is the 2-by-2 identity matrix.

The variance $\text{var}_v = \sigma(v,i)^2$ is estimated using an exponential averaging filter, modified for variable sampling rate

$$\text{var}_v_{\text{hat}} = \beta * \sigma(v,i-1)^2 + (1-\beta) * z(i)^2$$

$$\beta = (1-\alpha)^{(30/(1000 * f_{\text{max}}))}$$

where $f_{\max} = \max \{1/(T(j) - T(j-1))\}$ for j in $i-K+1 \dots i$ is the highest rate at which frames have been captured by the camera the last K frames and α is a filter coefficient typically chosen as a number in the interval $[0.1, 0.001]$. Since our assumption that $v(i)$ should be zero mean WGN is less accurate in some cases, we have introduced an additional outlier filter around the updates of var_v_hat . If $z(i) > 3 \text{ var_v_hat}$ the filter is updated with $3 \sqrt{\text{var_v_hat}}$ rather than $z(i)$. For instance $v(i)$ will not be white in situations where packets are sent at a higher rate than the channel capacity, in which case they will be queued behind each other. In a similar way, $Q(i)$ is chosen as a diagonal matrix with main diagonal elements given by

$$\text{diag}(Q(i)) = 30/(1000 * f_{\max})[10^{-10} \ 10^{-2}]^T$$

It is necessary to scale these filter parameters with the frame rate to make the detector respond as quickly at low frame rates as at high frame rates.

3.3. Over-use detector

The offset estimate $m(i)$ is compared with a threshold γ_1 . An estimate above the threshold is considered as an indication of over-use. Such an indication is not enough for the detector to signal over-use to the rate control subsystem. Not until over-use has been detected for at least γ_2 milliseconds and at least γ_3 frames, a definitive over-use will be signaled. However, if the offset estimate $m(i)$ was decreased in the last update, over-use will not be signaled even if all the above conditions are met. Similarly, the opposite state, under-use, is detected when $m(i) < -\gamma_1$. If neither over-use nor under-use is detected, the detector will be in the normal state.

3.4. Rate control

The rate control at the receiving side is designed to increase the receive-side estimate of the available bandwidth A_{hat} as long as the detected state is normal. Doing that assures that we, sooner or later, will reach the available bandwidth of the channel and detect an over-use.

As soon as over-use has been detected the receive-side estimate of the available bandwidth is decreased. In this way we get a recursive and adaptive estimate of the available bandwidth.

In this document we make the assumption that the rate control subsystem is executed periodically and that this period is constant.

The rate control subsystem has 3 states: Increase, Decrease and Hold. "Increase" is the state when no congestion is detected; "Decrease" is the state where congestion is detected, and "Hold" is a state that waits until built-up queues have drained before going to "increase" state.

The state transitions (with blank fields meaning "remain in state") are:

State ---->	Hold	Increase	Decrease
Signal-----			
v			
Over-use	Decrease	Decrease	
Normal	Increase		Hold
Under-use		Hold	Hold

The subsystem starts in the increase state, where it will stay until over-use or under-use has been detected by the detector subsystem. On every update the receive-side estimate of the available bandwidth is increased with a factor which is a function of the global system response time and the estimated measurement noise variance var_v_hat . The global system response time is the time from an increase that causes over-use until that over-use can be detected by the over-use detector. The variance var_v_hat affects how responsive the Kalman filter is, and is thus used as an indicator of the delay inflicted by the Kalman filter.

$$A_{\text{hat}}(i) = \eta * A_{\text{hat}}(i-1)$$

$$\eta(\text{RTT}, \text{var_v_hat}) = \frac{1.001+B}{1+e^{(b(d*\text{RTT} - (c1 * \text{var_v_hat} + c2)))}}$$

Here, B, b, d, c1 and c2 are design parameters.

Since the system depends on over-using the channel to verify the current available bandwidth estimate, we must make sure that our estimate doesn't diverge from the rate at which the sender is actually sending. Thus, if the sender is unable to produce a bit stream with the bit rate the receiver is asking for, the available bandwidth estimate must stay within a given bound. Therefore we introduce a threshold

$$A_hat(i) < 1.5 * R_hat(i)$$

where $R_hat(i)$ is the incoming bit rate measured over a T seconds window:

$$R_hat(i) = 1/T * \sum(L(j)) \text{ for } j \text{ from } 1 \text{ to } N(i)$$

$N(i)$ is the number of frames received the past T seconds and $L(j)$ is the payload size of frame j .

When an over-use is detected the system transitions to the decrease state, where the receive-side available bandwidth estimate is decreased to a factor times the currently incoming bit rate.

$$A_hat(i) = \alpha * R_hat(i)$$

α is typically chosen to be in the interval $[0.8, 0.95]$.

When the detector signals under-use to the rate control subsystem, we know that queues in the network path are being emptied, indicating that our available bandwidth estimate is lower than the actual available bandwidth. Upon that signal the rate control subsystem will enter the hold state, where the receive-side available bandwidth estimate will be held constant while waiting for the queues to stabilize at a lower level - a way of keeping the delay as low as possible. This decrease of delay is wanted, and expected, immediately after the estimate has been reduced due to over-use, but can also happen if the cross traffic over some links is reduced. In either case we want to measure the highest incoming rate during the under-use interval:

$$R_max = \max\{R_hat(i)\} \text{ for } i \text{ in } 1..K$$

where K is the number of frames of under-use before returning to the normal state. R_max is a measure of the actual bandwidth available and is a good guess of what bit rate the sender should be able to transmit at. Therefore the receive-side available bandwidth estimate will be set to R_max when we transition from the hold state to the increase state.

One design decision is when to send rate control messages. The time from a change in congestion to the sending of the feedback message is a limitation on how fast the sender can react. Sending too many messages giving no new information is a waste of bandwidth - but in the case of severe congestion, feedback messages can be lost, resulting in a failure to react in a timely manner.

The conclusion is that feedback messages should be sent on a "heartbeat" schedule, allowing the sender side control to react to missing feedback messages by reducing its send rate, but they should also be sent whenever the estimated bandwidth value has changed significantly, without waiting for the heartbeat time, up to some limiting upper bound on the send rate.

The minimum interval is named `t_min_fb_interval`.

The maximum interval is named `t_max_fb_interval`.

The permissible values of these intervals will be bounded by the RTP session's RTCP bandwidth and its `rtcp_frr` setting.

[TODO: Get some example values for these timers]

4. Sender side control

An additional congestion controller resides at the sending side. It bases its decisions on the round-trip time, packet loss and available bandwidth estimates transmitted from the receiving side.

The available bandwidth estimates produced by the receiving side are only reliable when the size of the queues along the channel are large enough. If the queues are very short, over-use will only be visible through packet losses, which aren't used by the receiving side algorithm.

This algorithm is run every time a receive report arrives at the sender, which will happen no more often than `t_min_fb_interval`, and no less often than `t_max_fb_interval`. If no receive report is received within $2 \times t_{\text{max_fb_interval}}$ (indicating at least 2 lost feedback reports), the algorithm will take action as if all packets in the interval have been lost, resulting in a halving of the send rate.

- o If 2-10% of the packets have been lost since the previous report from the receiver, the sender available bandwidth estimate $As(i)$ (As denotes 'sender available bandwidth') will be kept unchanged.
- o If more than 10% of the packets have been lost a new estimate is calculated as $As(i) = As(i-1)(1-0.5p)$, where p is the loss ratio.
- o As long as less than 2% of the packets have been lost $As(i)$ will be increased as $As(i) = 1.05(As(i-1) + 1000)$

The new send-side estimate is limited by the TCP Friendly Rate

Control formula [RFC3448] and the receive-side estimate of the available bandwidth $A(i)$:

$$As(i) \geq \frac{8s}{R\sqrt{2bp/3} + (t_{RTO}(3\sqrt{3bp/8}) * p * (1+32p^2))}$$

$$As(i) \leq A(i)$$

where b is the number of packets acknowledged by a single TCP acknowledgement (set to 1 per TFRC recommendations), t_{RTO} is the TCP retransmission timeout value in seconds (set to $4R$) and s is the average packet size in bytes. R is the round-trip time in seconds.

(The multiplication by 8 comes because TFRC is computing bandwidth in bytes, while this document computes bandwidth in bits.)

In words: The sender-side estimate will never be larger than the receiver-side estimate, and will never be lower than the estimate from the TFRC formula.

We motivate the packet loss thresholds by noting that if the transmission channel has a small amount of packet loss due to over-use, that amount will soon increase if the sender does not adjust his bit rate. Therefore we will soon enough reach above the 10 % threshold and adjust $As(i)$. However if the packet loss rate does not increase, the losses are probably not related to self-induced channel over-use and therefore we should not react on them.

5. Interoperability Considerations

There are three scenarios of interest, and one included for reference

- o Both parties implement the algorithms described here
- o Sender implements the algorithm described in section Section 4, recipient does not implement Section 3
- o Recipient implements the algorithm in section Section 3, sender does not implement Section 4.

In the case where both parties implement the algorithms, we expect to see most of the congestion control response to slowly varying conditions happen by TMMBR/REMB messages from recipient to sender. At most times, the sender will send less than the congestion-inducing bandwidth limit C , and when he sends more, congestion will be detected before packets are lost.

If sudden changes happen, packets will be lost, and the sender side control will trigger, limiting traffic until the congestion becomes low enough that the system switches back to the receiver-controlled state.

In the case where sender only implements, we expect to see somewhat higher loss rates and delays, but the system will still be overall TCP friendly and self-adjusting; the governing term in the calculation will be the TFRC formula.

In the case where recipient implements this algorithm and sender does not, congestion will be avoided for slow changes as long as the sender understands and obeys TMMBR/REMB; there will be no backoff for packet-loss-inducing changes in capacity. Given that some kind of congestion control is mandatory for the sender according to the TMMBR spec, this case has to be reevaluated against the specific congestion control implemented by the sender.

6. Implementation Experience

This algorithm has been implemented in the open-source WebRTC project.

7. Further Work

This draft is offered as input to the congestion control discussion.

Work that can be done on this basis includes:

- o Consideration of timing info: It may be sensible to use the proposed TFRC RTP header extensions [I-D.gharai-avtcore-rtp-tfrc] to carry per-packet timing information, which would both give more data points and a timestamp applied closer to the network interface. This draft includes consideration of using the transmission time offset defined in [RFC5450]
- o Considerations of cross-channel calculation: If all packets in multiple streams follow the same path over the network, congestion or queueing information should be considered across all packets between two parties, not just per media stream. A feedback message (REMB) that may be suitable for such a purpose is given in [I-D.alvestrand-rmcat-remb].
- o Considerations of cross-channel balancing: The decision to slow down sending in a situation with multiple media streams should be taken across all media streams, not per stream.

- o Considerations of additional input: How and where packet loss detected at the recipient can be added to the algorithm.
- o Considerations of locus of control: Whether the sender or the recipient is in the best position to figure out which media streams it makes sense to slow down, and therefore whether one should use TMMBR to slow down one channel, signal an overall bandwidth change and let the sender make the decision, or signal the (possibly processed) delay info and let the sender run the algorithm.
- o Considerations of over-bandwidth estimation: Whether we can use the estimate of how much we're over bandwidth in section 3 to influence how much we reduce the bandwidth, rather than using a fixed factor.
- o Startup considerations. It's unreasonable to assume that just starting at full rate is always the best strategy.
- o Dealing with sender traffic shaping, which delays sending of packets. Using send-time timestamps rather than RTP timestamps may be useful here, but as long as the sender's traffic shaping does not spread out packets more than the bottleneck link, it should not matter.
- o Stability considerations. It is not clear how to show that the algorithm cannot provide an oscillating state, either alone or when competing with other algorithms / flows.

These are matters for further work; since some of them involve extensions that have not yet been standardized, this could take some time.

8. IANA Considerations

This document makes no request of IANA.

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

9. Security Considerations

An attacker with the ability to insert or remove messages on the connection will, of course, have the ability to mess up rate control, causing people to send either too fast or too slow, and causing congestion.

In this case, the control information is carried inside RTP, and can be protected against modification or message insertion using SRTP, just as for the media. Given that timestamps are carried in the RTP header, which is not encrypted, this is not protected against disclosure, but it seems hard to mount an attack based on timing information only.

10. Acknowledgements

Thanks to Randell Jesup, Magnus Westerlund, Varun Singh, Tim Panton, Soo-Hyun Choo, Jim Gettys, Ingemar Johansson, Michael Welzl and others for providing valuable feedback on earlier versions of this draft.

11. References

11.1. Normative References

- [I-D.alvestrand-rmcat-remb]
Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", draft-alvestrand-rmcat-remb-00 (work in progress), January 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.

11.2. Informative References

- [I-D.gharai-avtcore-rtp-tfrc]
Gharai, L. and C. Perkins, "RTP with TCP Friendly Rate Control", draft-gharai-avtcore-rtp-tfrc-01 (work in

progress), September 2011.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

Appendix A. Change log

A.1. Version -00 to -01

- o Added change log
- o Added appendix outlining new extensions
- o Added a section on when to send feedback to the end of section 3.3 "Rate control", and defined min/max FB intervals.
- o Added size of over-bandwidth estimate usage to "further work" section.
- o Added startup considerations to "further work" section.
- o Added sender-delay considerations to "further work" section.
- o Filled in acknowledgements section from mailing list discussion.

A.2. Version -01 to -02

- o Defined the term "frame", incorporating the transmission time offset into its definition, and removed references to "video frame".
- o Referred to "m(i)" from the text to make the derivation clearer.
- o Made it clearer that we modify our estimates of available bandwidth, and not the true available bandwidth.
- o Removed the appendixes outlining new extensions, added pointers to REMB draft and RFC 5450.

Authors' Addresses

Henrik Lundin
Google
Kungsbron 2
Stockholm 11122
Sweden

Stefan Holmer
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: holmer@google.com

Harald Alvestrand (editor)
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: harald@alvestrand.no

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 5, 2012

R. Jesup
Mozilla
H. Alvestrand
Google
March 4, 2012

Congestion Control Requirements For Real Time Media
draft-jesup-rtp-congestion-reqs-00

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 by low-delay, semi-reliable data delivery, are different from the requirements for bulk transfer like FTP or bursty transfers like Web pages, and the TCP algorithms are not suitable for this traffic.

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.

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].

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 <http://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 September 5, 2012.

Copyright Notice

Copyright (c) 2012 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 (<http://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. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements	3
3. IANA Considerations	6
4. Security Considerations	6
5. Acknowledgements	6
6. References	7
6.1. Normative References	7
6.2. Informative References	7
Authors' Addresses	7

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 fast as possible" mode, 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 useless.

One particular protocol portfolio 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 YouTube-like 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 low-delay transit for real-time traffic, even when faced with intermediate bottlenecks and competing flows.
 - A. It should also deal well with routing changes and interface changes (WiFi to 3G data, etc) which may radically change the bandwidth available.

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.
 - A. 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.
3. The algorithm should 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.
 - A. If possible, it should also share information and adaptation with other non-RTP flows between the same endpoints, such as a WebRTC data channel
4. The algorithm should not require any special support from network elements (ECN, etc). As much as possible, it should leverage existing information about the incoming flows to provide feedback to the sender. Examples of this information are the packet arrival times, packet timestamps, packet sizes, packet losses. 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.
 - A. When signals such as ECN are available, it is good if they can be utilized.
5. Since the assumption here is a set of RTP streams, the backchannel typically should be done via RTCP; the alternative would be to include it in a reverse RTP channel using header extensions.
 - A. In order to react sufficiently quickly, the AVPF/SAVPF RTP profile[RFC4585] 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 allowing a higher maximum percentage for RTCP data.

- C. Note that RTCP is of course unreliable
 - D. Bandwidth for the feedback messages should be minimized (such as via RFC 5506 [RFC5506]) to allow RTCP without SR/RR)
 - E. Header extensions would avoid the RTCP timing rules issues, and allow the application to allocate bandwidth as needed for the congestion algorithm.
 - F. 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.
- 6. It should attempt to avoid bandwidth 'collapse' when facing a long-lived saturating TCP flow or flows. (I.e. a classic delay-sensitive algorithm will reduce bandwidth to keep delay down until the TCP flow has all the bandwidth). See the Cx-TCP algorithm discussed in a recent Transactions On Networking [cx-tcp] for an example of a delay-sensitive congestion-control algorithm that transitions to a loss-based mode when competing with TCP flows - at the cost of increased delay.
 - 7. The algorithm should be stable and low-delay when faced with active queue management (AQM) in the channel.
 - 8. The algorithm should quickly adapt to initial network conditions at the start of a flow; the adaptation may be faster than adaptation later in a flow. This should occur both if the initial bandwidth is above or below the bottleneck bandwidth.
 - A. 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!").
 - 9. Where possible, the algorithm should leverage and piggyback on other RTCP communications, such as SR/RR, rctp-fb PLI, RPSI, SLI or application-specific NACK messages (such as for loss information).

10. It should be evaluated in how it works both with backbone-router bottlenecks, (asymmetric) local-loop bottlenecks, and local-lan (WiFi/etc) bottlenecks.
11. 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.
12. It should be stable if the RTP streams are halted or discontinuous (VAD/DTX); after a resumption of RTP data it may adapt more quickly (similar to the start of a flow).

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 perceived available bandwidth are conceivable, 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-00 (work in progress), June 2011.
- [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.

6.2. Informative References

- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [cx-tcp] Budzisz, L., Stanojevic, R., Schlote, A., Baker, F., and R. Shorten, "On the Fair Coexistence of Loss- and Delay-Based TCP", December 2011.

Authors' Addresses

Randell Jesup
Mozilla
USA

Email: randell-ietf@jesup.org

Harald Alvestrand
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: harald@alvestrand.no

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 10, 2013

P. O'Hanlon
University of Oxford
K. Carlberg
Gll
July 9, 2012

Congestion control algorithm for lower latency and lower loss media
transport
draft-ohanlon-rmcat-dflow-00

Abstract

This memo provides an initial design for a congestion control algorithm, for media transport, which aims to provide for lower delay and lower loss communications.

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 <http://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 January 10, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://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. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions, Definitions and Acronyms	3
3. Background	4
3.1. TFRC	4
3.2. Delay-Based schemes	5
4. Objectives	6
5. Design Outline	6
5.1. Congestion Detection	7
6. Further Work	7
7. IANA Considerations	7
8. Security Considerations	8
9. References	8
9.1. Normative References	8
9.2. Informative References	8
Authors' Addresses	8

1. Introduction

This memo outlines DFlow, a congestion control algorithm that aims to minimise delay and loss by using delay-based techniques. The scheme is based upon TCP Friendly Rate Control (TFRC) [RFC5348], and adds a delay-based congestion detection scheme which feeds into a 'congestion event history' mechanism based upon TFRC's loss history. This then provides for a 'congestion event rate' which drives the TCP equation.

Low delay congestion control is important for real-time streams as high delay can render the communication unacceptable [ITU.G114.2003]. Unfortunately on today's Internet many paths have an excess of buffering which can lead to persistent high latencies, which has become known as the Bufferbloat phenomenon. These problems are particularly apparent with loss-based congestion control schemes such as TCP, as they operate by filling the queues on a path till loss occurs, thus maximising the delay. The unfortunate consequence is that loss-based approaches not only lead to high delay for their own packets but also introduce delays and losses for all other flows that traverse those filled queues.

Thus when competing with TCP, without the widespread deployment of Active Queue Management, it is not possible to maintain low delay as TCP will do its best to keep the queues full and maximise the delay. Furthermore when competing with TCP other flows will incur losses which should be used to operate a loss-based algorithm whilst in the presence of the TCP flows.

However there are a many paths where the flows are not competing directly with TCP and where delay may be minimised.

The DFlow scheme can transport media with low delay and loss on paths where there is no direct competition with TCP in the same queue. Though we are currently testing some techniques to enable it compete with loss-based schemes (at the expense of delay) but they will be included in a later version of the draft. In simulations it has been seen to be reasonably fair when competing with other DFlow streams.

2. Conventions, Definitions and Acronyms

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 RFC 2119 [RFC2119].

3. Background

Whilst the existing standard for media transport, Real-time Transport Protocol (RTP) [RFC3550], suggest that congestion control should be employed, in practice many systems tend to use fixed or variable bit rate UDP and do very little or no adaptation to their network environment. Most of the existing work on realtime congestion control algorithms has been rooted in TCP-friendly approaches but with smoother adaptation cycles. TCP congestion control is unsuitable for interactive media for a number of reasons including the fact that it is loss-based so it maximises the latency on a path, it changes its transmit rate to quickly for multimedia, and favours reliability over timeliness. Various TCP-friendly congestion control algorithms such as TFRC [RFC5348], Sislem's LDA+, and Choi's TFWC have been devised for unreliable media transport, that attempt to smooth the short-term variation in sending rate. More recently there have been development of some delay-based schemes which aim to provide for low delay.

3.1. TFRC

TFRC is a rate based receiver driven congestion control algorithm which utilises the Padhye TCP equation to provide a TCP-friendly rate. The sender explicitly sets the transmission rate, using the TCP equation driven by the loss event rate which is measured and fed back by the receiver, where a loss event consists of one or more packet losses within a single RTT. It utilises a weighted smoothed loss event rate, and EWMA smoothed RTT, as input to the TCP Equation which enables it to achieve a smoother rate adaptation that provides for a more suitable transport for multimedia. TFRC was primarily aimed at streaming media delivery where a smooth rate and TCP-friendliness are more important than low latency operation.

However there are number of issues with TFRC as regards media transport:

Loss-based operation: Firstly since it is a loss-based based scheme the latency is maximised which is a problem for real-time transport over heavily buffered paths. The other problem with loss-based protocols is that they rely on a certain level of packet loss which can be an issue for media traffic since retransmissions are problematic. This problem is more of a concern as rates decrease since the TCP equation requires a corresponding increase in loss rates.

Bursty media flows: Many media flows exhibit bursty behaviour due to a number of factors. Firstly there may be negative bursts (i.e. gaps) due to silence or low motion which can lead oscillatory behaviours due to the its data-limited and/or idle behaviours. Secondly there may be postive bursts (i.e. larger than normal) can also be due to the bursty nature of the media and codec (e.g. I-frames) which can be lead to drops or increased latency. Whilst the current version of TFRC [RFC5348] has attempted to address these issues, they are still a concern.

Small RTTs environments: When operating in low RTT environments (<5ms), such as a LAN, TFRC has can have problems with scheduling packet transmissions as interpacket timings can be lower than application level clock granularity. Furthermore these conditions can Whilst the current version of TFRC [RFC5348] has attempted to address these issues, they can still be a concern in low RTT environments.

Variable packet sizes: As originally designed TFRC will only operate correctly when packet sizes are close to MTU size, and when the packet sizes are much smaller fairness issues arise. Although there have been attempts to address this problem for small packets [RFC4828] it is not clear how to deal with flows that do vary their packet sizes substantially. Though this issue is only really a marked problem with lower bit rate video flows or variable packet rate audio.

3.2. Delay-Based schemes

In the last few years there has been a renewed interest in the use of delay based congestion control for media, with a slightly different emphasis to that of the history of TCP approaches such as Jain's CARD, Crowcroft and Jon's Tri-S, Brakmo's Vegas, and more recently Tan et al's Compound TCP. Whilst the goal with these media based transports is to actually minimise the latency of the flow, as opposed to just using delay as an early indication of loss. This is of particular relevance on paths with large queues, as is the case with a number of today's Internet paths. In 2007 Ghanbari et al did some pioneering work on delay-based video congestion control using fuzzy logic based systems. Recently there has been on going activity in the IETF as part of the Low Extra Delay Background Transport (LEDBAT) Working Group which aims to provide a less than best effort delay-based transport with lower delay. More recently Google published a contribution to the Real-Time Communication in WEB-browsers (RTCWeb) Working Group, which has now been spun off to the new BOF on RTP Media Congestion Avoidance Techniques (rmcat).

4. Objectives

The objectives of DFlow are to provide for low delay and low loss media transport when possible. We also aim to provide (in a future version of the draft) mechanisms to provide for better burst management, and loss-mode operation (the key being the switch from loss-mode back to delay-mode).

Lower Delay: The one-way delay should be kept well within the acceptable levels of 150ms, and MUST NOT exceed 400ms [ITU.G114.2003].

Lower Loss: For media transport is important to minimise loss as it is usually not possible to retransmit within the delay budget for many connections. Whilst modern codecs can tolerate some loss it is beneficial to avoid it. The benefit of low delay congestion control is that since it aims to operate within the queuing boundaries it generally avoids loss.

Smoothness: The media rate should aim to be smooth within the constraints of the media, codec, and the network path. A smooth rate generally provides for a more palatable media consumption.

Fairness: The system should aim to be reasonably fair with TCP flows and itself. Initially we aim for self fairness and we will aim to tackle TCP fairness when we have sufficiently robust loss-mode operation.

[Burst Management]: [Due in later rev] We aim to work on mechanisms to manage the bursty nature of media allowing it maintain a smoother quality. A smooth rate generally provides for a more palatable media consumption.

[Loss-based mode]: [Due in later rev] We aim to work on mechanisms to allow the system to compete with loss-based congestion control and maintain throughput, though without additional network support it is understood that the delay (and loss) would be largely beyond control.

5. Design Outline

At this stage DFlow utilises the core aspects of TFRC, such as its rate based operation, utilisation of the TCP equation, and its smoother rate. It also utilises similar packet contents and signalling mechanisms. However as the design evolves we realise that DFlow may become quite separate from TFRC.

5.1. Congestion Detection

The delay-based detection algorithm operates by sampling the one-way delay (OWD) of each packet arrival by comparing the send timestamp with the receive time. The OWD is sampled over a small set time period, `sample_period`, (typically 100ms) and the minima stored as the `BaseDelay` over a longer period, `base_period`, (typically 5-10xRTT). The current OWD is then compared to the `BaseDelay` and if it exceeds a set threshold, `CDthresh`, (typically around 50ms) then the packet is considered for the next stage of detection. The following test is based upon the gradient of the delay change over two `sample_periods`, indicating that delay is on the increase, if it is positive then a 'congestion event' is logged. The minima of the OWDs are used to reduce noise of the measurements, which is beneficial in the case of variable link types such as Wifi.

```
If ((BaseDelayMinNxRTT - OWD) > CDthresh AND
    DelayIncreaseOver2RTTs > 0)
    DelayCongestionEvent = True
```

Figure 1

This algorithm then provides input to the 'congestion interval history' (or TFRC's 'loss interval history') mechanism, which is combined with normal input from the TFRC packet loss detection mechanisms, from which a 'congestion event rate' is derived which is then fed into the TCP equation to determine the send rate.

Note that we disable TFRC's oscillation reduction mechanism from Section 4.5 [RFC5348] as it adversely affects the delay-based operation.

We have performed some simulations of the above mechanism in operation and have found it to be reasonable fair to itself, though it is not as smooth as TFRC.

6. Further Work

The design is still under development and there is clearly more work to be done. But we are seeking feedback on these ideas and future directions.

7. IANA Considerations

This document makes no requests of IANA.

8. Security Considerations

With a congestion control algorithm an attacker can attempt to interfere with the protocol to cause rate changes. However encryption of the protocol will protect it against such threats.

9. References

9.1. Normative References

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

9.2. Informative References

- [ITU.G114.2003] International Telecommunications Union, "One-way transmission time", ITU-T Recommendation G.707, May 2003.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

Authors' Addresses

Piers O'Hanlon
University of Oxford
Oxford Internet Institute
1 St Giles
Oxford OX1 3JS
United Kingdom

Email: piers.ohanlon@oii.ox.ac.uk

Ken Carlberg
G11
1600 Clarendon Blvd
Arlington VA
USA

Email: carlberg@g11.org.uk

