Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 25, 2013

H. Lundin
S. Holmer
H. Alvestrand, Ed.
Google

October 22, 2012

A Google Congestion Control Algorithm for Real-Time Communication on the World Wide Web

draft-alvestrand-rtcweb-congestion-03

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 as an input document to the RMCAT working group on congestion control for media streams. The mailing list of that WG is rmcat@ietf.org.

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 April 25, 2013.

Copyright Notice

Table of Contents

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:

- 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

- 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

- 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:

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

X_hat 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:

- RTP packet - an RTP packet containing media data.
- 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.
- Incoming media streams - a stream of frames consisting of RTP packets.
- Media codec - has a bandwidth control, and encodes the incoming media stream into an RTP stream.
- RTP sender - sends the RTP stream over the network to the RTP

receiver. Generates the RTP timestamp.

- RTP receiver - receives the RTP stream, notes the time of arrival. Regenerates the media stream for the recipient.
- RTCP sender at RTP sender - sends sender reports with mappings between RTP timestamps and NTP time.
- RTCP sender at RTP receiver - sends receiver reports and TMMBR/ REMB messages.
- RTCP receiver at RTP sender - receives receiver reports and TMMBR/ REMB messages, reports these to sender side control.
- RTCP receiver at RTP receiver.
- Sender side control - takes loss rate info, round trip time info, and TMMBR/REMB messages and computes a sending bitrate.

- 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 four parts: an RTP timestamp to NTP time conversion, arrival-time filter, an over-use detector, and a remote rate-control.

3.1. Procsesing multiple streams using RTP timestamp to NTP time conversion

It is common that multiple RTP streams are sent from the sender to the receiver. In such a situation the RTP timestamps of incoming can first be converted to a common time base using the RTP timestamp and NTP time pairs in RTCP SR reports[RFC3550]. The converted timestamps can then be used instead of RTP timestamps in the arrival-time filtering, and since all streams from the same sender have timestamps

in the same time base they can all be processed by the same filter. This has the advantage of quicker reactions and reduces problems of noisy measurements due to self-inflicted cross-traffic.

In the time interval from the start of the call until a stream from the same sender has received an RTCP SR report, the receiver-side control operates in single-stream mode. In that mode only one RTP stream can be processed by the over-use detector. As soon as a stream has received one or more RTCP SR reports the receiver-side control can change to a multi-stream mode, where all RTP streams from the same sender which have received one or more RTCP SR reports can be processed by the over-use detector. When switching to the multi-stream mode the state of the over-use detector must be modified to avoid a time base mismatch. This can either be done by resetting the stored RTP timestamp values or by converting them using the newly received RTCP SR report.

3.2. 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 ts to send a frame of size L over a path with a capacity of C is roughly

$$ts = L/C$$

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.

3.3. 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

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

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

$$\bar{\theta}(i+1) = \bar{\theta}(i) + \bar{u}(i)$$

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

Equation 7

$$Q(i) = E\{\bar{u}(i)\,\bar{u}(i)^T\}$$

Given equation 5 we get

Equation 8

$$d(i) = \bar{h}(i)^T \bar{\theta}(i) + v(i)$$

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

where v(i) is zero mean white Gaussian measurement noise with variance var_v = sigma(v,i)^2

The Kalman filter recursively updates our estimate

$$\hat{\theta}(i) = [1/\hat{C}(i) \quad \hat{m}(i)]^T$$

as

```
  z(i) = d(i) - h_bar(i)^T * theta_hat(i-1)

  theta_hat(i) = theta_hat(i-1) + z(i) * k_bar(i)

                            E(i-1) * h_bar(i)
  k_bar(i) = ------------------------------------------
              var_v_hat + h_bar(i)^T * E(i-1) * h_bar(i)

  E(i) = (I - K_bar(i) * h_bar(i)^T) * E(i-1) + Q(i)
```

I is the 2-by-2 identity matrix.

The variance var_v = sigma(v,i)^2 is estimated using an exponential averaging filter, modified for variable sampling rate

```
  var_v_hat = beta*sigma(v,i-1)^2 + (1-beta)*z(i)^2

  beta = (1-alpha)^(30/(1000 * f_max))
```

where f_max = max {1/(T(j) - T(j-1))} for j in i-K+1...i is the highest rate at which frames have been captured by the camera the last K frames and alpha 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 var_v_hat the filter is updated with 3 sqrt(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

```
  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.

3.4. Over-use detector

The offset estimate m(i) is compared with a threshold gamma_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 gamma_2 milliseconds and at least gamma_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.

3.5. 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_hat(i) = eta*A_hat(i-1)
                                  1.001+B
eta(RTT, var_v_hat) = ----------------------------------------
                           1+e^(b(d*RTT - (c1 * 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)) for j from 1 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. Ideally T should be chosen to match the rate controller at the sender. A window between 0.5 and 1 second is recommended.

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)

alpha 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)} for i 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 2x t_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.

- 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.

- 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.

- 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):

$$As(i) >= \frac{8\ s}{R*sqrt(2*b*p/3) + (t\_RTO*(3*sqrt(3*b*p/8) * p * (1+32*p^2)))}$$

$$As(i) <= A(i)$$

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 4*R) 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

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

- 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]

- 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].

- 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.

- Considerations of additional input: How and where packet loss detected at the recipient can be added to the algorithm.

- 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.

- 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.

- Startup considerations. It's unreasonable to assume that just starting at full rate is always the best strategy.

- 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-01 (work in progress), July 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

- Added change log
- Added appendix outlining new extensions
- Added a section on when to send feedback to the end of section 3.3 "Rate control", and defined min/max FB intervals.

- Added size of over-bandwidth estimate usage to "further work" section.
- Added startup considerations to "further work" section.
- Added sender-delay considerations to "further work" section.
- Filled in acknowledgements section from mailing list discussion.

A.2. Version -01 to -02

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

A.3. Version -02 to -03

- Added a section on how to process multiple streams in a single estimator using RTP timestamps to NTP time conversion.
- Stated in introduction that the draft is aimed at the RMCAT working group.

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