Network Working Group	H. Lundin
Internet-Draft	S. Holmer
Intended status: Informational	H. T. Alvestrand, Ed.
Expires: March 08, 2012	Google
	September 05, 2011

A Google Congestion Control for Real-Time Communication on the World Wide Web
draft-alvestrand-rtcweb-congestion-00

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

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 accomodate 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], standard RTCP feedback reports and Temporary Maximum Media Stream Bit Rate Requests (TMMBR) as defined in [RFC5104] section 3.5.4.

1.1. Mathemathical 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.

2. System model

The following elements are in the system:

*Incoming media stream

*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.

*RTCP sender at RTP receiver - sends receiver reports and TMMBR messages.

*RTCP receiver at RTP sender - receives receiver reports and TMMBR 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 messages and computes a sending bitrate.

*Receiver side control - takes the packet arrival info at the RTP receiver and decides when to send TMMBR 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 filter

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 video 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

$$ts = L/C$$

we can model the inter-arrival time as

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

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 of w(i) to make it 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.

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).

The parameters d(i) and dL(i) are readily available for each frame i, and we want to estimate C 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

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

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

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

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

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

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

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). 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.

3.2. 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.

3.3. Rate control

The rate control at the receiving side is designed to increase the available bandwidth estimate 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 available bandwidth estimate is decreased. In this way we get a recursive and adaptive estimate of the available bandwidth.

In this design description 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 ----> / Signal v	Hold	Increase	Decrease
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 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.

$$A(i) = eta * A(i-1)$$

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

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.

When an over-use is detected the system transitions to the decrease state, where the 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 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 we should be able to transmit at. Therefore the available bandwidth will be set to Rmax when we transition from the hold state to the increase state.

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 [[how often do we expect? and why?]]. If no receive report is recieved within [[what timeout?]], the algorithm will take action as if all packets in the interval have been lost. [[does that make sense?]]

*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.

(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 we have small amount of packet losses due to over-use, that amount will soon increase if we don't adjust our 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 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; 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.

* 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.

* 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.

These are matters for further work; since some of them involve extensions that have not yet been standardized, this could take some time, and it's important to consider when this work can be completed.

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

11. References

11.1. Normative References

[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.

11.2. Informative References

[I-D.gharai-avtcore-rtp-tfrc]	Gharai, L and C Perkins, "RTP with TCP Friendly Rate Control", Internet-Draft draft-gharai-avtcore-rtp-tfrc-01, September 2011.
[RFC2914]	Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

Authors' Addresses

Henrik Lundin Lundin Google Kungsbron 2 Stockholm, 11122 Sweden

Stefan Holmer Holmer Google Kungsbron 2 Stockholm, 11122 Sweden
EMail: holmer@google.com

Harald Alvestrand editor Alvestrand Google Kungsbron 2
Stockholm, 11122 Sweden EMail: harald@alvestrand.no