

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: August 18, 2014

H. Lundin  
S. Holmer  
Google  
L. De Cicco  
S. Mascolo  
Politecnico di Bari  
H. Alvestrand, Ed.  
Google  
February 14, 2014

A Google Congestion Control Algorithm for Real-Time Communication  
draft-alvestrand-rmcat-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 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](mailto: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 August 18, 2014.

## Copyright Notice

Copyright (c) 2014 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. Procseing multiple streams using RTP timestamp to NTP time conversion . . . . .	5
3.2. Arrival-time model . . . . .	5
3.3. Arrival-time filter . . . . .	7
3.4. Over-use detector . . . . .	8
3.5. Rate control . . . . .	10
4. Sender side control . . . . .	12
5. Interoperability Considerations . . . . .	14
6. Implementation Experience . . . . .	14
7. Further Work . . . . .	14
8. IANA Considerations . . . . .	16
9. Security Considerations . . . . .	16
10. Acknowledgements . . . . .	16
11. References . . . . .	16
11.1. Normative References . . . . .	16
11.2. Informative References . . . . .	17
Appendix A. Change log . . . . .	17
A.1. Version -00 to -01 . . . . .	17
A.2. Version -01 to -02 . . . . .	18
A.3. Version -02 to -03 . . . . .	18
A.4. rtcweb-03 to rmcat-00 . . . . .	18
A.5. rmcat -00 to -01 . . . . .	18
A.6. rmcat -01 to -02 . . . . .	18
Authors' Addresses . . . . .	18

## 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 signaling 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-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  $z$ .

$X_{\text{bar}}^T$  The transpose of vector  $X_{\text{bar}}$ .

$E\{X\}$  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 with mappings between RTP timestamps and NTP time.
- 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 four parts: an RTP timestamp to NTP time conversion, arrival-time filter, an over-use detector, and a remote rate-control.

#### 3.1. Processing 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  $t_s$  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  $dL = 0$ .

### 3.3. 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

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

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

$$\begin{aligned}
z(i) &= d(i) - h\_bar(i)^T * \theta\_hat(i-1) \\
\theta\_hat(i) &= \theta\_hat(i-1) + z(i) * k\_bar(i) \\
k\_bar(i) &= \frac{E(i-1) * h\_bar(i)}{\text{var\_v\_hat} + h\_bar(i)^T * E(i-1) * h\_bar(i)} \\
E(i) &= (I - K\_bar(i) * h\_bar(i)^T) * E(i-1) + Q(i)
\end{aligned}$$

$I$  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

$$\begin{aligned}
\text{var\_v\_hat} &= \beta * \sigma(v,i-1)^2 + (1-\beta) * z(i)^2 \\
\beta &= (1-\alpha)^{(30/(1000 * f\_max))}
\end{aligned}$$

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  $\alpha$  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  $\text{var\_v\_hat}$ . 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.

### 3.4. Over-use detector

The offset estimate  $m(i)$  is compared with a threshold  $\gamma_1(i)$ . 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  $m(i) < -\text{gamma\_1}(i)$ . If neither over-use nor under-use is detected, the detector will be in the normal state.

The threshold  $\text{gamma\_1}$  has a remarkable impact on the overall dynamics and performance of the algorithm. In particular, it has been shown that when using a static threshold  $\text{gamma\_1}$ , a flow controlled by the proposed algorithm can be starved by a concurrent TCP flow [Pv13]. This starvation can be avoided by increasing the threshold  $\text{gamma\_1}$  to a sufficiently large value.

The reason is that, by using a larger value of  $\text{gamma\_1}$ , a larger queuing delay can be tolerated, whereas with a small  $\text{gamma\_1}$ , the over-use detector quickly reacts to a small increase in the offset estimate  $m(i)$  by generating an over-use signal that reduces  $A_r$ . Thus, it is necessary to dynamically tune the threshold  $\text{gamma\_1}$  to get good performance in the most common scenarios, such as when competing with loss-based flows.

For this reason, we propose to vary the threshold  $\text{gamma\_1}(i)$  according to the following dynamic equation:

$$\text{gamma\_1}(i) = \text{gamma\_1}(i-1) + (t(i)-t(i-1)) * K(i) * (|m(i)| - \text{gamma\_1}(i-1))$$

with  $K(i)=K_d$  if  $|m(i)| < \text{gamma\_1}(i-1)$  or  $K(i)=K_u$  otherwise. The rationale is to increase  $\text{gamma\_1}(i)$  when  $m(i)$  is outside of the range  $[-\text{gamma\_1}(i-1), \text{gamma\_1}(i-1)]$ , whereas, when the offset estimate  $m$  falls back into the range,  $\text{gamma\_1}$  is decreased. In this way when  $m(i)$  quickly increases, for instance due to a TCP flow entering the same bottleneck,  $\text{gamma\_1}(i)$  increases and avoids the uncontrolled generation of over-use signals which may lead to starvation of the flow controlled by the proposed algorithm [Dec13].

On the other hand, when  $m(i)$  falls back into the range  $[-\text{gamma\_1}(i-1), \text{gamma\_1}(i-1)]$  the threshold  $\text{gamma\_1}(i)$  is decreased so that a lower queuing delay can be achieved.

We suggest to choose  $K_u > K_d$  so that the rate at which  $\text{gamma\_1}$  is increased is higher than the rate at which it is decreased. With this setting it is possible to quickly increase the threshold in the case of a concurrent TCP flow and prevent starvation.

### 3.5. Rate control

The rate control at the receiving side is designed to increase the receive-side estimate of the available bandwidth  $A_{\text{hat}}$  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  $\text{var}_v\text{hat}$ . 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  $\text{var}_v\text{hat}$  affects how responsive the Kalman filter is, and is thus used as an indicator of the delay inflicted by the Kalman filter.

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

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:

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

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

$$R_{\max} = \max\{R_{\text{hat}}(i)\} \text{ for } i \text{ in } 1..K$$

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  $A(i)$ :

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

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

where  $b$  is the number of packets acknowledged by a single TCP acknowledgment (set to 1 per TFRC recommendations),  $t_{\text{RTO}}$  is the TCP retransmission timeout value in seconds (set to  $4 \times R$ ) 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

- 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 queuing 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-03 (work in progress), October 2013.
- [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.
- [Pv13] De Cicco, L., Carlucci, G., and S. Mascolo, "Understanding the Dynamic Behaviour of the Google Congestion Control", Packet Video Workshop , December 2013.
- [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 acknowledgments 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.

## A.3. Version -02 to -03

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

## A.4. rtcweb-03 to rmcat-00

Renamed draft to link the draft name to the RMCAT WG.

## A.5. rmcat -00 to -01

Spellcheck. Otherwise no changes, this is a "keepalive" release.

## A.6. rmcat -01 to -02

- o Added Luca De Cicco and Saverio Mascolo as authors.
- o Extended the "Over-use detector" section with new technical details on how to dynamically tune the offset `gamma_1` for improved fairness properties.
- o Added a reference to a paper analyzing the behavior of the proposed algorithm.

## Authors' Addresses

Henrik Lundin  
Google  
Kungsbron 2  
Stockholm 11122  
Sweden

Stefan Holmer  
Google  
Kungsbron 2  
Stockholm 11122  
Sweden

Email: holmer@google.com

Luca De Cicco  
Politecnico di Bari  
Via Orabona, 4  
Bari 70125  
Italy

Email: l.decicco@poliba.it

Saverio Mascolo  
Politecnico di Bari  
Via Orabona, 4  
Bari 70125  
Italy

Email: mascolo@poliba.it

Harald Alvestrand (editor)  
Google  
Kungsbron 2  
Stockholm 11122  
Sweden

Email: harald@alvestrand.no

RMCAT WG  
Internet-Draft  
Intended status: Informational  
Expires: January 26, 2015

V. Singh  
J. Ott  
Aalto University  
July 25, 2014

Evaluating Congestion Control for Interactive Real-time Media  
draft-ietf-rmcat-eval-criteria-02

Abstract

The Real-time Transport Protocol (RTP) is used to transmit media in telephony and video conferencing applications. This document describes the guidelines to evaluate new congestion control algorithms for interactive point-to-point real-time media.

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 26, 2015.

Copyright Notice

Copyright (c) 2014 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 . . . . .	2
2. Terminology . . . . .	3
3. Metrics . . . . .	3
3.1. RTP Log Format . . . . .	4
4. Guidelines . . . . .	5
4.1. Avoiding Congestion Collapse . . . . .	5
4.2. Stability . . . . .	5
4.3. Media Traffic . . . . .	5
4.4. Start-up Behaviour . . . . .	6
4.5. Diverse Environments . . . . .	6
4.6. Varying Path Characteristics . . . . .	6
4.7. Reacting to Transient Events or Interruptions . . . . .	6
4.8. Fairness With Similar Cross-Traffic . . . . .	7
4.9. Impact on Cross-Traffic . . . . .	7
4.10. Extensions to RTP/RTCP . . . . .	7
5. Minimum Requirements for Evaluation . . . . .	7
6. Status of Proposals . . . . .	7
7. Security Considerations . . . . .	8
8. IANA Considerations . . . . .	8
9. Contributors . . . . .	8
10. Acknowledgements . . . . .	8
11. References . . . . .	8
11.1. Normative References . . . . .	8
11.2. Informative References . . . . .	9
Appendix A. Application Trade-off . . . . .	10
A.1. Measuring Quality . . . . .	10
Appendix B. Change Log . . . . .	10
B.1. Changes in draft-ietf-rmcat-eval-criteria-02 . . . . .	10
B.2. Changes in draft-ietf-rmcat-eval-criteria-01 . . . . .	10
B.3. Changes in draft-ietf-rmcat-eval-criteria-00 . . . . .	10
B.4. Changes in draft-singh-rmcat-cc-eval-04 . . . . .	10
B.5. Changes in draft-singh-rmcat-cc-eval-03 . . . . .	11
B.6. Changes in draft-singh-rmcat-cc-eval-02 . . . . .	11
B.7. Changes in draft-singh-rmcat-cc-eval-01 . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

This memo describes the guidelines to help with evaluating new congestion control algorithms for interactive point-to-point real time media. The requirements for the congestion control algorithm are outlined in [I-D.ietf-rmcat-cc-requirements]). This document builds upon previous work at the IETF: Specifying New Congestion Control Algorithms [RFC5033] and Metrics for the Evaluation of Congestion Control Algorithms [RFC5166].

The guidelines proposed in the document are intended to help prevent a congestion collapse, promote fair capacity usage and optimize the media flow's throughput. Furthermore, the proposed algorithms are expected to operate within the envelope of the circuit breakers defined in [I-D.ietf-avtcore-rtp-circuit-breakers].

This document only provides broad-level criteria for evaluating a new congestion control algorithm and the working group should expect a thorough scientific study to make its decision. The results of the evaluation are not expected to be included within the internet-draft but should be cited in the document.

## 2. Terminology

The terminology defined in RTP [RFC3550], RTP Profile for Audio and Video Conferences with Minimal Control [RFC3551], RTCP Extended Report (XR) [RFC3611], Extended RTP Profile for RTCP-based Feedback (RTP/AVPF) [RFC4585] and Support for Reduced-Size RTCP [RFC5506] apply.

## 3. Metrics

[RFC5166] describes the basic metrics for congestion control. Metrics that are of interest for interactive multimedia are:

- o Throughput.
- o Minimizing oscillations in the transmission rate (stability) when the end-to-end capacity varies slowly.
- o Delay.
- o Reactivity to transient events.
- o Packet losses and discards.
- o Section 2.1 of [RFC5166] discusses the tradeoff between throughput, delay and loss.

Each experiment is expected to log every incoming and outgoing packet (the RTP logging format is described in Section 3.1). The logging can be done inside the application or at the endpoints using pcap (packet capture, e.g., tcpdump, wireshark). The following are calculated based on the information in the packet logs:

1. Sending rate, Receiver rate, Goodput
2. Packet delay

3. Packet loss
4. If using, retransmission or FEC: residual loss
5. Packets discarded from the playout or de-jitter buffer
6. Fairness or Unfairness: Experiments testing the performance of an RMCAT proposal against any cross-traffic must define its expected criteria for fairness. The "unfairness" test guideline (measured at 1s intervals) is:
  1. Does not trigger the circuit breaker.
  2. No RMCAT stream achieves more than 3 times the average throughput of the RMCAT stream with the lowest average throughput, for a case when the competing streams have similar RTTs.
  3. RTT should not grow by a factor of 3 for the existing flows when a new flow is added.For example, see the test scenarios described in [I-D.sarker-rmcat-eval-test].
7. Convergence time: The time taken to reach a stable rate at startup, after the available link capacity changes, or when new flows get added to the bottleneck link.
8. Bandwidth Utilization, defined as ratio of the instantaneous sending rate to the instantaneous bottleneck capacity. This metric is useful when an RMCAT flow is by itself or competing with similar cross-traffic.

From the logs the statistical measures (min, max, mean, standard deviation and variance) for the whole duration or any specific part of the session can be calculated. Also the metrics (sending rate, receiver rate, goodput, latency) can be visualized in graphs as variation over time, the measurements in the plot are at 1 second intervals. Additionally, from the logs it is possible to plot the histogram or CDF of packet delay.

[Open issue (1): Using Jain-fairness index (JFI) for measuring self-fairness between RTP flows? measured at what intervals? visualized as a CDF or a timeseries? Additionally: Use JFI for comparing fairness between RTP and long TCP flows? ]

### 3.1. RTP Log Format

The log file is tab or comma separated containing the following details:

Send or receive timestamp (unix)  
RTP payload type  
SSRC  
RTP sequence no  
RTP timestamp  
marker bit  
payload size

If the congestion control implements, retransmissions or FEC, the evaluation should report both packet loss (before applying error-resilience) and residual packet loss (after applying error-resilience).

#### 4. Guidelines

A congestion control algorithm should be tested in simulation or a testbed environment, and the experiments should be repeated multiple times to infer statistical significance. The following guidelines are considered for evaluation:

##### 4.1. Avoiding Congestion Collapse

The congestion control algorithm is expected to take an action, such as reducing the sending rate, when it detects congestion. Typically, it should intervene before the circuit breaker [I-D.ietf-avtcore-rtp-circuit-breakers] is engaged.

Does the congestion control propose any changes to (or diverge from) the circuit breaker conditions defined in [I-D.ietf-avtcore-rtp-circuit-breakers].

##### 4.2. Stability

The congestion control should be assessed for its stability when the path characteristics do not change over time. Changing the media encoding rate estimate too often or by too much may adversely affect the application layer performance.

##### 4.3. Media Traffic

The congestion control algorithm should be assessed with different types of media behavior, i.e., the media should contain idle and data-limited periods. For example, periods of silence for audio, varying amount of motion for video, or bursty nature of I-frames.

The evaluation may be done in two stages. In the first stage, the endpoint generates traffic at the rate calculated by the congestion controller. In the second stage, real codecs or models of video

codecs are used to mimic application-limited data periods and varying video frame sizes.

#### 4.4. Start-up Behaviour

The congestion control algorithm should be assessed with different start-rates. The main reason is to observe the behavior of the congestion control in different test scenarios, such as when competing with varying amount of cross-traffic or how quickly does the congestion control algorithm achieve a stable sending rate.

#### 4.5. Diverse Environments

The congestion control algorithm should be assessed in heterogeneous environments, containing both wired and wireless paths. Examples of wireless access technologies are: 802.11, GPRS, HSPA, or LTE. One of the main challenges of the wireless environments for the congestion control algorithm is to distinguish between congestion induced loss and transmission (bit-error) loss. Congestion control algorithms may incorrectly identify transmission loss as congestion loss and reduce the media encoding rate by too much, which may cause oscillatory behavior and deteriorate the users' quality of experience. Furthermore, packet loss may induce additional delay in networks with wireless paths due to link-layer retransmissions.

#### 4.6. Varying Path Characteristics

The congestion control algorithm should be evaluated for a range of path characteristics such as, different end-to-end capacity and latency, varying amount of cross traffic on a bottleneck link and a router's queue length. For the moment, only DropTail queues are used. However, if new Active Queue Management (AQM) schemes become available, the performance of the congestion control algorithm should be again evaluated.

In an experiment, if the media only flows in a single direction, the feedback path should also be tested with varying amounts of impairments.

The main motivation for the previous and current criteria is to identify situations in which the proposed congestion control is less performant.

#### 4.7. Reacting to Transient Events or Interruptions

The congestion control algorithm should be able to handle changes in end-to-end capacity and latency. Latency may change due to route updates, link failures, handovers etc. In mobile environment the

end-to-end capacity may vary due to the interference, fading, handovers, etc. In wired networks the end-to-end capacity may vary due to changes in resource reservation.

#### 4.8. Fairness With Similar Cross-Traffic

The congestion control algorithm should be evaluated when competing with other RTP flows using the same or another candidate congestion control algorithm. The proposal should highlight the bottleneck capacity share of each RTP flow.

#### 4.9. Impact on Cross-Traffic

The congestion control algorithm should be evaluated when competing with standard TCP. Short TCP flows may be considered as transient events and the RTP flow may give way to the short TCP flow to complete quickly. However, long-lived TCP flows may starve out the RTP flow depending on router queue length.

The proposal should also measure the impact on varied number of cross-traffic sources, i.e., few and many competing flows, or mixing various amounts of TCP and similar cross-traffic.

#### 4.10. Extensions to RTP/RTCP

The congestion control algorithm should indicate if any protocol extensions are required to implement it and should carefully describe the impact of the extension.

### 5. Minimum Requirements for Evaluation

The minimal requirements for RMCAT proposals is to produce or present results for the test scenarios described in Section 5 of [I-D.sarker-rmcat-eval-test] (Basic Test Cases).

### 6. Status of Proposals

Congestion control algorithms are expected to be published as "Experimental" documents until they are shown to be safe to deploy. An algorithm published as a draft should be experimented in simulation, or a controlled environment (testbed) to show its applicability. Every congestion control algorithm should include a note describing the environments in which the algorithm is tested and safe to deploy. It is possible that an algorithm is not recommended for certain environments or perform sub-optimally for the user.

[Editor's Note: Should there be a distinction between "Informational" and "Experimental" drafts for congestion control algorithms in RMCAT.

[RFC5033] describes Informational proposals as algorithms that are not safe for deployment but are proposals to experiment with in simulation/testbeds. While Experimental algorithms are ones that are deemed safe in some environments but require a more thorough evaluation (from the community).]

## 7. Security Considerations

Security issues have not been discussed in this memo.

## 8. IANA Considerations

There are no IANA impacts in this memo.

## 9. Contributors

The content and concepts within this document are a product of the discussion carried out in the Design Team.

Michael Ramalho provided the text for a specific scenario, which is now covered in [I-D.sarker-rmcat-eval-test].

## 10. Acknowledgements

Much of this document is derived from previous work on congestion control at the IETF.

The authors would like to thank Harald Alvestrand, Anna Brunstrom, Luca De Cicco, Wesley Eddy, Lars Eggert, Kevin Gross, Vinayak Hegde, Stefan Holmer, Randell Jesup, Karen Nielsen, Piers O'Hanlon, Colin Perkins, Michael Ramalho, Zaheduzzaman Sarker, Timothy B. Terriberry, Michael Welzl, and Mo Zanaty for providing valuable feedback on earlier versions of this draft. Additionally, also thank the participants of the design team for their comments and discussion related to the evaluation criteria.

## 11. References

### 11.1. Normative References

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [I-D.ietf-rmcat-cc-requirements]  
Jesup, R., "Congestion Control Requirements For RMCAT", draft-ietf-rmcat-cc-requirements-02 (work in progress), February 2014.
- [I-D.ietf-avtcore-rtp-circuit-breakers]  
Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", draft-ietf-avtcore-rtp-circuit-breakers-05 (work in progress), February 2014.

## 11.2. Informative References

- [RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, August 2007.
- [RFC5166] Floyd, S., "Metrics for the Evaluation of Congestion Control Mechanisms", RFC 5166, March 2008.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [I-D.sarker-rmcat-eval-test]  
Sarker, Z., Singh, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-sarker-rmcat-eval-test-00 (work in progress), February 2014.
- [SA4-EVAL]  
R1-081955, 3GPP., "LTE Link Level Throughput Data for SA4 Evaluation Framework", 3GPP R1-081955, 5 2008.
- [SA4-LR]  
S4-050560, 3GPP., "Error Patterns for MBMS Streaming over UTRAN and GERAN", 3GPP S4-050560, 5 2008.

[TCP-eval-suite]

Lachlan, A., Marcondes, C., Floyd, S., Dunn, L., Guillier, R., Gang, W., Eggert, L., Ha, S., and I. Rhee, "Towards a Common TCP Evaluation Suite", Proc. PFLDnet. 2008, August 2008.

## Appendix A. Application Trade-off

Application trade-off is yet to be defined. see RMCAT requirements [I-D.ietf-rmcat-cc-requirements] document. Perhaps each experiment should define the application's expectation or trade-off.

### A.1. Measuring Quality

No quality metric is defined for performance evaluation, it is currently an open issue. However, there is consensus that congestion control algorithm should be able to show that it is useful for interactive video by performing analysis using a real codec and video sequences.

## Appendix B. Change Log

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

### B.1. Changes in draft-ietf-rmcat-eval-criteria-02

- o Incorporated fairness test as a working test.
- o Updated text on minimum evaluation requirements.

### B.2. Changes in draft-ietf-rmcat-eval-criteria-01

- o Removed Appendix B.
- o Removed Section on Evaluation Parameters.

### B.3. Changes in draft-ietf-rmcat-eval-criteria-00

- o Updated references.
- o Resubmitted as WG draft.

### B.4. Changes in draft-singh-rmcat-cc-eval-04

- o Incorporate feedback from IETF 87, Berlin.
- o Clarified metrics: convergence time, bandwidth utilization.

- o Changed fairness criteria to fairness test.
- o Added measuring pre- and post-repair loss.
- o Added open issue of measuring video quality to appendix.
- o clarified use of DropTail and AQM.
- o Updated text in "Minimum Requirements for Evaluation"

B.5. Changes in draft-singh-rmcat-cc-eval-03

- o Incorporate the discussion within the design team.
- o Added a section on evaluation parameters, it describes the flow and network characteristics.
- o Added Appendix with self-fairness experiment.
- o Changed bottleneck parameters from a proposal to an example set.
- o

B.6. Changes in draft-singh-rmcat-cc-eval-02

- o Added scenario descriptions.

B.7. Changes in draft-singh-rmcat-cc-eval-01

- o Removed QoE metrics.
- o Changed stability to steady-state.
- o Added measuring impact against few and many flows.
- o Added guideline for idle and data-limited periods.
- o Added reference to TCP evaluation suite in example evaluation scenarios.

Authors' Addresses

Varun Singh  
Aalto University  
School of Electrical Engineering  
Otakaari 5 A  
Espoo, FIN 02150  
Finland

Email: [varun@comnet.tkk.fi](mailto:varun@comnet.tkk.fi)  
URI: <http://www.netlab.tkk.fi/~varun/>

Joerg Ott  
Aalto University  
School of Electrical Engineering  
Otakaari 5 A  
Espoo, FIN 02150  
Finland

Email: [jo@comnet.tkk.fi](mailto:jo@comnet.tkk.fi)

RMCAT WG  
Internet-Draft  
Intended status: Informational  
Expires: December 28, 2014

I. Johansson  
Z. Sarker  
Ericsson AB  
June 26, 2014

Self-Clocked Rate Adaptation for Multimedia  
draft-johansson-rmcat-scream-cc-02

Abstract

This memo describes a rate adaptation framework for conversational video services. The solution conforms to the packet conservation principle and uses a hybrid loss and delay based congestion control algorithm. The framework is evaluated over both simulated bottleneck scenarios as well as in a LTE (Long Term Evolution) system simulator and is shown to achieve both low latency and high video throughput in these scenarios.

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 December 28, 2014.

Copyright Notice

Copyright (c) 2014 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 . . . . .	2
1.1. Wireless (LTE) access properties . . . . .	3
2. Terminology . . . . .	3
3. The adaptation framework . . . . .	3
3.1. Congestion control . . . . .	7
3.2. Transmission scheduling . . . . .	8
3.3. Media rate control . . . . .	8
4. Conclusion . . . . .	8
5. Open issues . . . . .	9
6. Acknowledgements . . . . .	9
7. IANA Considerations . . . . .	9
8. Security Considerations . . . . .	9
9. Change history . . . . .	9
10. Algorithm details . . . . .	10
10.1. Sender side functions . . . . .	10
10.1.1. RTP packet queue handling (a.k.a Sender queue) . . . . .	10
10.1.2. Transmission scheduler . . . . .	11
10.1.3. Reception of RTCP feedback in sender . . . . .	13
10.1.4. Congestion window adjustment . . . . .	15
10.1.5. Video encoder . . . . .	16
10.1.6. Video encoder rate adaptation . . . . .	17
10.2. Receiver side functions . . . . .	19
10.2.1. Reception of RTP packet . . . . .	19
11. Simulations . . . . .	19
11.1. DL . . . . .	21
11.1.1. RTT 40ms 3km/h . . . . .	21
11.1.2. RTT 40ms 30km/h . . . . .	22
11.2. UL . . . . .	22
11.2.1. RTT 40ms 3km/h . . . . .	23
11.2.2. RTT 40ms 30km/h . . . . .	23
12. References . . . . .	24
12.1. Normative References . . . . .	24
12.2. Informative References . . . . .	24
Authors' Addresses . . . . .	25

## 1. Introduction

Rate adaptation is considered as an important part of a interactive realtime communication as the transmission channel bandwidth may vary over period of time. Wireless access such as LTE (Long Term Evolution), which is an integral part of the current Internet, increases the importance of rate adaptation as the channel bandwidth of a default LTE bearer [QoS-3GPP] can change considerably in a very

short time frame. Thus a rate adaptation solution for interactive realtime media, such as WebRTC, in LTE must be both quick and also able to operate over a large span in available channel bandwidth. This memo describes a solution that borrows the self-clocking principle of TCP and combines it with a new delay based rate adaptation algorithm, LEDBAT [RFC6817]. Because neither TCP nor LEDBAT was designed for interactive realtime media, a few extra features are needed to make the concept work well with in this context. This memo describes these extra features.

### 1.1. Wireless (LTE) access properties

[I-D.draft-sarker-rmcat-cellular-eval-test-cases] introduces the complications that can be observed in wireless environments. Wireless access such as LTE can typically not guarantee a given bandwidth, this holds for true especially for default bearers. The network throughput may vary considerably for instance in cases where the wireless terminal is moving around.

Unlike wireline bottlenecks with large statistical multiplexing it is not possible to try to maintain a given bitrate when congestion is detected with the hope that other flows will yield, this because there are generally few other flows competing for the same bottleneck. Each user gets its own variable throughput bottleneck, where the throughput depends on factors like channel quality, load and historical throughput. The bottom line is thus, if the throughput drops, the sender has no other option than to reduce the bitrate. In addition, the grace time, i.e. allowed reaction time from the time that the congestion is detected until a reaction in terms of a rate reduction is effected, is generally very short, in the order of one RTT (Round Trip Time).

## 2. Terminology

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 RFC2119 [RFC2119]

## 3. The adaptation framework

The adaptation framework has similarities to concepts like TFWC [TFWC]. One important property is self-clocking and compliance to the packet conservation principle. The packet conservation principle is described as an important key-factor behind the protection of networks from congestion [FACK].

The packet conservation principle is realized by including a vector of the sequence numbers of received packets in the feedback from the

receiver back to the sender, the sender keeps a list of transmitted packets and their respective sizes. This information is then used to determine how many bytes can be transmitted. A congestion window puts an upper limit on how many bytes can be in flight, i.e. transmitted but not yet acknowledged. The congestion window is determined in a way similar to LEDBAT. This ensures that the e2e latency is kept low. The basic functionality is quite simple, there are however a few steps to take to make the concept work with conversational media. These will be briefly described in sections Section 3.1 to Section 3.3.

The feedback is over RTCP [RFC3550] and is based on [RFC4585]. It is implemented as a transport layer feedback message, see proposed example in Figure 1. The feedback control information part (FCI) consists of the following elements.

- o Timestamp: A timestamp value indicating when the last packet was received which makes it possible to compute the one way (extra) delay (OWD).
- o The ACK list (Highest received sequence number + ACK vector): Makes it possible to detect lost packets and determine the number of bytes in flight.
- o ECN (Explicit Congestion Notification) echo: Makes it possible to indicate if packets are ECN-CE (ECN Congestion Experienced) marked. The use for the 8 ECN echo bits is T.B.D.
- o Source quench bit (Q): Makes it possible to request the sender to reduce its congestion window. This is useful if WebRTC media is received from many hosts and it becomes necessary to balance the bitrates between the streams. The exact behavior and use for the source quench bit is T.B.D.

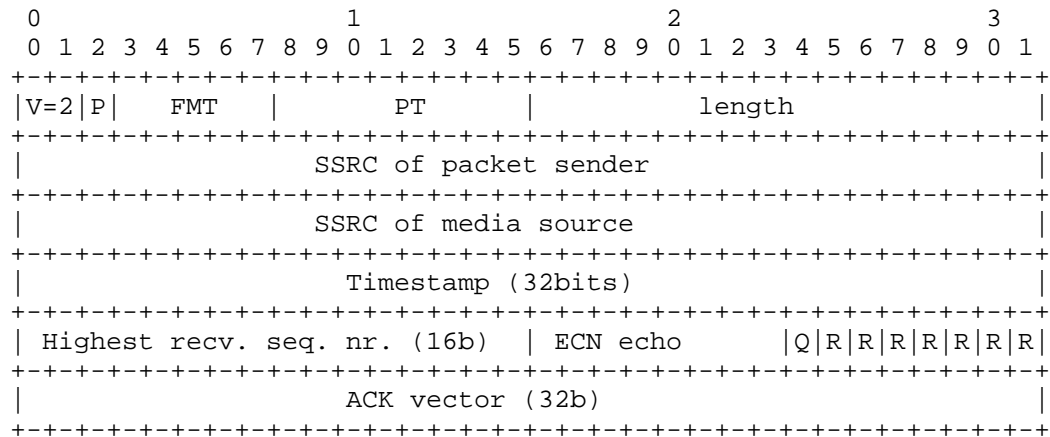


Figure 1: Transport layer feedback message

To make the feedback as frequent as possible, the feedback packets are transmitted as reduced size RTCP according to [RFC5506].

The timestamp clock time base is typically set to the same time base as the media source in question but as the protocol described here is not dependent on the media it can be set to a fixed value defined in this specification. The ACK vector is here a bit vector that indicates the reception of the last  $1+32 = 33$  RTP packets. The ACK vector may also be RLE coded.

Section 10 describes the main algorithm details and how the feedback is used.

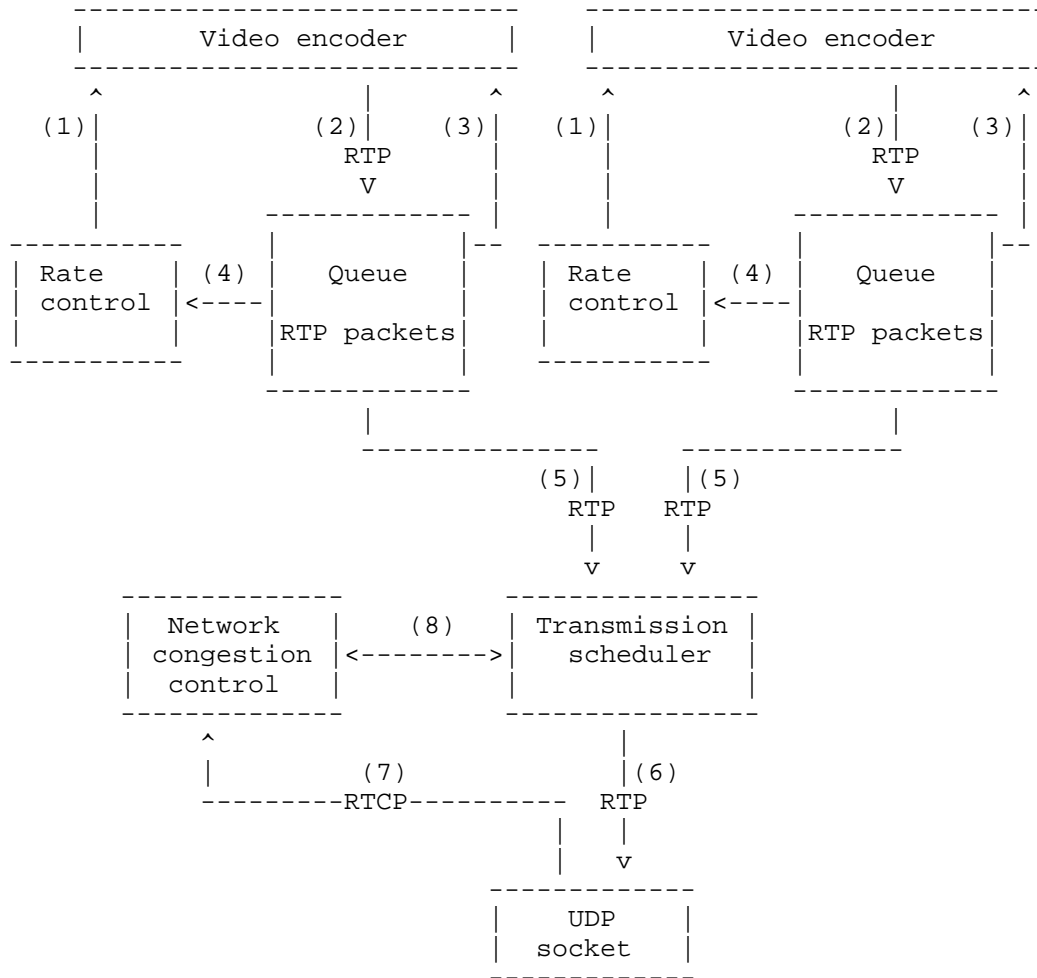


Figure 2: Rate adaptation framework

Figure 2 shows the functional overview of the adaptation framework. Each media type or source implements rate control and a queue, where encoded media frames are temporarily stored for transmission, the figure shows the details for one two video sources. Video frames are encoded and forwarded to the queue (2). The media rate adaptation adapts to the age of the oldest RTP frame in the queue and controls the video bitrate (1). It is also possible to make the video encoder skip frames and thus temporarily reduce the frame rate if the queue age exceeds a given threshold (3). The RTP packets are picked from each queue based on some defined priority order or simply in a round robin fashion (5). A transmission scheduler takes care of the transmission of RTP packets, to be written to the UDP socket (6). In the general case all media must go through the packet scheduler and is allowed to be transmitted if the number of bytes in flight is less than the congestion window. However audio frames can be allowed to be transmitted as audio is typically low bitrate and thus contributes little to congestion, this is however something that is left as an implementation choice. RTCP packets are received (7) and the information about bytes in flight and congestion window is exchanged between the network congestion control and the transmission scheduler (8).

The rate adaptation solution constitutes three parts; congestion control, transmission scheduling and media rate adaptation.

### 3.1. Congestion control

The congestion control sets an upper limit on how much data can be in the network (bytes in flight); this limit is called CWND (congestion window) and is used in the transmission scheduling.

A congestion control method, similar to LEDBAT, measures the OWD (one way delay). The congestion window is allowed to increase if the OWD is below a predefined target, otherwise the congestion window decreases. The delay target is typically set to 50-100ms. This ensures that the OWD is kept low on the average. The reaction to loss events is similar to that of loss based TCP, i.e. an instant reduction of CWND.

LEDBAT is designed with file transfers as main use case which means that the algorithm must be modified somewhat to work with rate-limited sources such as video. The modifications are:

- o Congestion window validation techniques. These are similar in action as the method described in [I-D.ietf-tcpm-newcwnd].
- o Fast start for bitrate increase. It makes the video bitrate ramp-up within 3 to 5 seconds. The behavior is similar to TCP

slowstart. The fast start is exited when the OWD exceeds a given threshold.

- o Adaptive delay target. This helps the congestion control to compete with FTP traffic to some degree.

### 3.2. Transmission scheduling

Transmission scheduling limits the output of data, given by the relation between the number of bytes in flight and the congestion window similar to TCP. Packet pacing is used to mitigate issues with coalescing that may cause increased jitter in the media traffic.

### 3.3. Media rate control

The media rate control serves to adjust the media bitrate to ramp up quickly enough to get a fair share of the system resources when link throughput increases.

The reaction to reduced throughput must be prompt in order to avoid getting too much data queued up in the sender frame queues. The queuing delay is determined and the media bitrate is decreased if it exceeds a threshold.

In cases where the sender frame queues increase rapidly such as the case of a RAT (Radio Access Type) handover it may be necessary to implement additional actions, such as discarding of encoded video frames or frame skipping in order to ensure that the sender frame queues are drained quickly. Frame skipping means that the frame rate is temporarily reduced. Discarding of old video frames is a more efficient way to reduce latency than frame skipping but it comes with a requirement to repair codec state.

## 4. Conclusion

This memo describes a congestion control framework for RMCAT that it is particularly good at handling the quickly changing condition in wireless network such as LTE. The solution conforms to the packet conservation principle and leverages on novel congestion control algorithms and recent TCP research, together with media bitrate determined by sender queuing delay and given delay thresholds. The solution has shown potential to meet the goals of high link utilization and prompt reaction to congestion. The solution is realized with a new RFC4585 transport layer feedback message.

## 5. Open issues

A list of open issues.

- o Describe how the RTCP feedback described in this memo is handled by mixers in various scenarios
- o Describe how clock drift compensation is done
- o RTCP AVPF mode. Determine if AVPF immediate mode is to prefer, see discussion in Section 11
- o Determine use of Q bit
- o Determine format and use of ECN echo field
- o The example code in Section 10 assumes a video source where the sizes of the video frames are scaled according to a scale-factor to produce the desired bitrate. This may not be implementable in a real-life encoder, hence the code in said section is tightly connected to mentioned synthetic video source model.

## 6. Acknowledgements

We would like to thank the following persons for their comments, questions and support during the work that led to this memo: Markus Andersson, Bo Burman, Tomas Frankkila, Laurits Hamm, Hans Hannu, Nikolas Hermanns, Stefan Haekansson, Erlendur Karlsson, Mats Nordberg, Jonathan Samuelsson, Rickard Sjoeborg, Magnus Westerlund.

## 7. IANA Considerations

A new RFC4585 transport layer feedback message needs to be standardized.

## 8. Security Considerations

The feedback can be vulnerable to attacks similar to those that can affect TCP. It is therefore recommended that the RTCP feedback is at least integrity protected.

## 9. Change history

A list of changes:

- o -01 to -02 : Updated GCC simulation results
- o -00 to -01 : Fixed a few bugs in example code

## 10. Algorithm details

This section describes the algorithm in a Java syntax. The code is not complete however and wont compile, for instance Java class constructors are omitted for brevity. Algorithm details that are missing are:

- o Fast start
- o Congestion Window Validation
- o Adjustment to competing flows
- o Discard old video frames in sender queue

### 10.1. Sender side functions

#### 10.1.1. RTP packet queue handling (a.k.a Sender queue)

The RTP packets produced by the video encoder are inserted in a FIFO queue. The FIFO order may be overridden for instance if retransmission of RTP packets is needed, in which case the RTP packet to be retransmitted is put first in queue.

```

/*
 * RtpPacket contains the RTP packet
 * Details are omitted
 */
class RtpPacket {

};

/*
 * The SenderQItem is a container for SSRC, timestamp when
 * item is added to the queue and the RTP packet itself
 */
class SenderQItem {
    int SSRC;
    int TS;
    RtpPacket rtpPacket;
};

/*
 * The list implements functions like
 * add(..) : add a SenderQItem
 * get()   : get the oldest Item
 * remove() : remove the oldest item
 * age()   : get the queuing delay of the oldest packet
 */
ArrayList<SenderQItem> senderQ = new list ArrayList<SenderQItem>;

/*
 * Add an RTP packet to the sender queue
 */
function addRtpPacket(int SSRC, RtpPacket rtpPacket) {
    senderQ.add(new SenderQItem(SSRC,now,rtpPacket));
}

```

#### 10.1.2. Transmission scheduler

The RTP packet transmission procedure is executed every  $tp$  interval where  $tp$  is computed according to the equation further down.

```

/*
 * The TransmitItem is a container for SSRC, timestamp when
 * item is transmitted to the queue and the RTP packet itself
 */
class TransmitItem {
    int SSRC;
    int TS_Tx;
    int SN;
    int size;
}

```

```
};

/*
 * The list implements functions like
 * add(..) : add a SenderQItem
 * get()    : get the oldest Item
 * remove() : remove the oldest item
 * age()    : get the queuing delay of the oldest packet
 * find(..) : find and return the item that matches....
 */
ArrayList<TransmitItem> transmitted = new ArrayList<TransmitItem>;

/*
 * Constants
 */
int MSS_INIT = 1200;

/*
 * Global variables
 */
int bytesInFlight = 0;
double tp = 0.001;
int mss = MSS_INIT;
int cwndMin = 2*mss;

/*
 * Function that is called every tp interval
 */
function tryTransmit() {
    RtpPacket rtpPacket = senderQ.get();
    if (bytesInFlight+rtpPacket.size < cwnd && rtpPacket != null) {
        /*
         * OK to transmit
         * Transmit next RTP packet in the sender queue and remove
         * the RTP packet from the sender queue
         */
        sendRtp(rtpPacket);
        senderQ.remove();

        /*
         * Add info about transmitted RTP packet to transmitted list
         * The RTP packet itself may be stored too to support
         * retransmission
         */
        transmitted.add(
            new TransmitItem(rtpPacket.SSRC,now,rtpPacket.SN,
                           rtpPacket.size));
    }
}
```

```
/*
 * update bytesInFlight
 */
bytesInFlight += rtpPacket.size;

/*
 * compute tp, we assume a min bw of 50kbps and a min tp of 1ms
 * for stable operation
 * this function implements the packet pacing
 */
bw = cwnd*8/rtt;
tp = max(0.001,max(rtpPacket.size*8/max(50000,bw)));

/*
 * Update MSS and cwndMin
 */
mss = max(mss,rtpPacket.size);
cwndMin = 2*mss;
} else {
/*
 * Not OK to transmit
 */
tp = 0.001;
}
}
```

#### 10.1.1.3. Reception of RTCP feedback in sender

From the RTCP feedback, the following information is used to adjust the number of bytesInFlight and to update the bytesNewlyAacked, the ackVector is also used to determine loss events. Furthermore TX\_Rx is used to determine owd.

```
/*
 * RtcpPacket contains the feedback RTCP packet
 * Details are omitted
 */
class RtcpPacket {

};

/*
 * Global variables
 */
double lastLossEvent = -1;
boolean lossEvent = false;
int bytesNewlyAacked = 0;
double owd = 0.0;
```

```
function rtcpFeedbackReceived(RtcpPacket rtcp) {
    item = transmitted.find(rtcp.SSRC, rtcp.SN)
    /*
     * Update OWD according to RFC6817
     * based on item.TS_Tx and rtcp.TS_Rx
     */
    updatedOwd(); // Function not specified in this memo

    /*
     * Update the number of bytesInFlight and bytesNewlyAacked
     * Remove items with sequence number lower than or equal to
     * rtcp.SN.
     * Sequence number wrap around is not considered in this code
     */
    for (TransmitItem item : transmitted) {
        if (item.SSRC == rtcp.SSRC && item.SN <= rtcp.SN) {
            bytesInFlight -= item.size;
            bytesNewlyAacked += item.size;
            /*
             * Remove item from transmitted list
             */
            transmitted.remove(item);
        }
    }
    /*
     * Determine if a loss event has occurred
     */
    if (now - lastLostEvent > rtt) {
        /*
         * A loss event is determined by a hole in the sequence
         * number space in the ACK vector, a guard time of the
         * last ACKed RTP SN is used to avoid false loss event
         * detection in the presence of packet reordering in the
         * network.
         */
        /*
         * Function loss event not specified in this memo
         */
        lossEvent = isLossEvent(rtcp.SN, rtcp.ackVector);

        if (lossEvent)
            lastLostEvent = now;
    }

    /*
     * Update the congestion window
     */
    updateCwnd();
}
```

```
}
```

#### 10.1.4. Congestion window adjustment

The congestion window is adjusted for every received RTCP feedback.

```
/*
 * Constants
 */
double cwndHeadroomMin = 1.0;
double cwndHeadroomMax = 2.0;
double gainUp = 1.0;
double gainDown = 1.0;
double beta = 0.8;
double OWD_TARGET = 0.08;

/*
 * Global variables
 */
double owdTarget = OWD_TARGET;
double lastOwd = 0.0;

function updateCwnd() {
  /*
   * offTarget is a normalized deviation from the owdTarget
   */
  offTarget = (owdTarget-owd)/owdTarget;

  /*
   * cwndHeadRoom gives indicates how much lower bytesInFlight
   * can be compared to cwnd to allow a cwnd increase
   */
  cwndHeadroom = cwndHeadroomMin+
    max(0.0,offTarget)*(cwndHeadroomMax-cwndHeadroomMin);

  if (lossEvent) {
    /*
     * loss event detected, decrease congestion window
     */
    cwnd = max(cwndMin, beta*cwnd);
    lossEvent = false;
  }

  if (offTarget > 0) {
    /*
     * owd is lower than owdTarget,
     * possible to increase cwnd
     */

```

```
    if bytesInFlight*cwndHeadroom > cwnd {
        /*
         * Pipe is sufficiently filled with data,
         * increase cwnd
         */
        cwnd += gainUp * offTarget *
            bytesNewlyAked * mss / cwnd;
    }
} else {
    if (owd-lastOwd >= 0.0) {
        /*
         * Decrease cwnd quickly if owd is constant high or
         * increasing
         */
        rttFactor = Math.min(2.0,Math.max(0.1, getRtt())/0.1);
        cwnd += gainDown * rttFactor * max(-3.0,offTarget) *
            bytesNewlyAked * mss / cwnd;
    } else {
        /*
         * Decrease cwnd slowly if owd is declining
         */
        cwnd += gainDown * max(-3.0,offTarget) *
            bytesNewlyAked * mss / cwnd;
    }
}
cwnd = max(cwndMin,cwnd);
lastOwd = owd;
bytesNewlyAked = 0;
}
```

#### 10.1.5. Video encoder

NOT\_SPECIFIED means that the values need to be specified based on video codec properties.

```
/*
 * Constants
 */
double scaleFactorMin = NOT_SPECIFIED_1;
double scaleFactorMax = NOT_SPECIFIED_2;
double framePeriod = NOT_SPECIFIED_3;

/*
 * Global variables
 */
boolean skipFrame = false;
double scaleFactor = scaleFactorMin;

/*
 * Function called for every grabbed video frame
 */
function encodeVideoFrame() {
    if (!skipFrame) {
        /*
         * Details of encode function call depends on
         * video encoder properties
         */
        rtpPacket = encode(..., scaleFactor);
        addRtpPacket(SSRC, rtpPacket);
    }
    adjustVideoBitrate();
}
```

#### 10.1.1.6. Video encoder rate adaptation

The video encoder rate is adjusted every RTT. The bitrate is controlled by a scale factor that is bounded by [minScaleFactor maxScaleFactor]. A history of the age of the oldest RTP packet in the sender queue over the 5 latest RTTs is maintained (ageHist).

```
/*
 * Constants
 */
int ageHistSizeMax = 5;

/*
 * Global variables
 */
double lastTimeAdjustVideoBitrate = 0.0;
ArrayList<double> ageHist = new ArrayList<double>;

/*
 * Function called every videoframe interval
```

```
*/
function adjustVideoBitrate() {
  if (senderQ.age > skipFrameTh)
    skipFrame = true;
  else
    skipFrame = false;
  if (now-lastTimeAdjustVideoBitrate < framePeriod)
    return;
  ageHist.add(senderQ.age());
  if (ageHist.size >= ageHistSizeMax) {
    age = ageHist.average(); // Compute average
    owdFraction = owd/owdTarget;
    if (age > framePeriod/2) {
      /*
       * Decrease the scale factor proportional to the age
       */
      scaleFactor = max(scaleFactorMin, scaleFactor*(1.0-age));
    } else {
      /*
       * Increase the scale factor
       */
      /*
       * Put an upper limit on how fast the scalefactor can
       * increase
       */
      scaleLimit = max(scaleFactor, scaleFactorMax*0.1)*0.2;
      /*
       * Increment is slowed down
       * if owd shows a tendency to increase
       */
      rampUpSlowDown = min(5.0, max(rampUpSlowDown*0.9,
        1.0+5.0*max(0.0,owdFraction-0.2)));

      increment =
        min(framePeriod*maxScaleFactor/
          (rampUpTime*rampUpSlowDown), scaleLimit);
      scaleFactor = min(maxScaleFactor, scaleFactor+increment);
    }
  }

  /*
   * Remove oldest element in age history
   */
  ageHist.remove();
}
```

## 10.2. Receiver side functions

### 10.2.1. Reception of RTP packet

An RTCP packet is created and scheduled for transmission. If an RTCP packet is already present and waiting for transmission, the new RTCP packet will replace the older RTCP feedback packet.

```
/*
 * Global variables
 */
RtcpPacket rtcpFeedbackPacket = null;
ArrayList<integer> rtpSn = new ArrayList<integer>;

/*
 * New RTP packet received
 */
function rtpReceived(rtp) {
    rtpSn.add(rtp.SN);

    /*
     * A new RTCP feedback is prepared for transmission,
     * due to RTCP bandwidth and timing rules it may happen that
     * an RTCP feedback has not been transmitted when a new
     * feedback packet is generated. To make feedback as timely
     * as possible, older unsent feedback packets should be replaced
     * by new feedback packets.
     */
    rtcpFeedbackPacket = RtcpFeedbackPacket(SSRC,now,rtp.SN,rtpSn)

    /*
     * Decode RTP packet and render media
     */
}
```

## 11. Simulations

A state of the art dynamic LTE simulation with settings according to Table 1 , derived from [I-D.draft-sarker-rmcat-cellular-eval-test-cases] was used to assess the performance of the algorithm. The simulator models the whole protocol chain including handover, radio propagation etc.

## LTE simulation configuration

Cellular layout	21 cells (7 sites); 3GPP case 1 settings
System setup	10MHz bandwidth, 2GHz carrier frequency, eNB transmission power 40W, MIMO transmission mode
Channel	Typical Urban
Propagation model	Okumura-Hata model
Scheduler	DL scheduler: Proportional fair. UL scheduler: Proportional fair
User generation	Poisson arrival based user arrival
Mobility	UE moves straight in a randomly selected direction, at a speed 3km/h or 30km/h, ideal handover model (no user plane interruption, no handover signaling)
Traffic scenario	Video: Rate adaptive video, codec : H.264, bitrate 150-1500kbps. RTCP BW: 75kbps. Audio: Frame size 20ms, bitrate 20kbps, frame skip feature enabled. FTP: 500kB objects corresponding to an average of 4Mbps load per cell for the DL test case and 2Mbps load per cell for the UL test case.

Table 1

The self-clocked algorithm is compared against the Google congestion control algorithm [I-D.alvestrand-rmcat-congestion]. The load level i.e. the intensity of new video users and the FTP load is:

- o DL simulations: Video users: 2 to 16 users/cell, FTP load: 4Mbps
- o UL simulations: Video users: 1 to 10 users/cell, FTP load: 2Mbps

The latency is expressed as "packet 98%ile, user 95%ile delay", meaning that the "98%ile delay" is determined for all users, i.e 98% of the video frames or IP packets have a delay less than the "packet 98%ile delay" value. Based on this the "user 95%ile delay" is determined, meaning that 95% of the users have a "packet 98%ile delay" less than said "user 95%ile value", this is a relatively strict metric but it gives a fairly good idea of how stable the video (and audio playback) is.

The users/cell indications and the associated performance metrics should not be treated as exact values as there is a lot of dependencies in propagation models, antenna configurations, scheduler implementations, other cross traffic involved. Therefore the tables

should be read as a comparison between congestion control algorithms at the same given network conditions. Only the results with AQM enabled are shown in the tables below.

In general the results indicate that SCReAM is more stable in terms of latency and packet loss than GCC. Furthermore, SCReAM achieves good throughput at low load levels in both uplink and downlink. SCReAM implements the frame skipping mechanism in these simulations, which means that the frame rate is reduced if the queuing delay in the sender queue exceeds 100ms. The 30km/h test cases are more challenging as the channel conditions vary more quickly, also in this case SCReAM performs better than GCC. The performance can be further improved if frames in the sender queue are discarded if they are too old to be rendered in a meaningful way in the receiver.

The SCReAM simulations was done in AVPF regular mode with an RTCP feedback overhead of ~12kbps (including IP and UDP). The effect of the AVPF regular mode is however that the RTCP feedback is not transmitted for each received IP packet but rather for each video frame and this can potentially cause a less stable self-clocking. AVPF immediate mode should be tried out to see if it gives an improvement.

#### 11.1. DL

##### 11.1.1. RTT 40ms 3km/h

GCC

Users/cell	2	4	6	8	12	16
Video tail latency [ms]	121	381	879	1026	1078	1182
IP packet tail latency [ms]	129	418	882	1027	1083	1228
Average PLR [%]	0.0	0.1	0.5	2.0	6.4	15.5
Average bitrate [kbps]	1153	857	605	462	306	210

Table 2

SCReAM

Users/cell	2	4	6	8	12	16
Video tail latency [ms]	126	192	192	258	412	559
IP packet tail latency [ms]	92	126	139	171	233	293
Average PLR [%]	0.0	0.0	0.0	0.1	0.2	0.8
Average bitrate [kbps]	1202	812	575	461	328	232

Table 3

## 11.1.2. RTT 40ms 30km/h

GCC

Users/cell	2	4	6	8	12	16
Video tail latency [ms]	524	905	1108	1373	2273	2099
IP packet tail latency [ms]	528	902	1094	1400	2299	2177
Average PLR [%]	0.1	0.5	1.8	4.3	15.8	34.6
Average bitrate [kbps]	941	523	309	217	108	67

Table 4

SCReAM

Users/cell	2	4	6	8	12	16
Video tail latency [ms]	229	263	350	414	748	1470
IP packet tail latency [ms]	134	164	197	224	438	920
Average PLR [%]	0.0	0.1	0.1	0.2	1.4	7.4
Average bitrate [kbps]	878	512	351	277	184	131

Table 5

## 11.2. UL

## 11.2.1. RTT 40ms 3km/h

GCC

Users/cell	1	2	3	5	8	10
Video tail latency [ms]	135	123	390	477	926	984
IP packet tail latency [ms]	147	129	352	473	929	982
Average PLR [%]	0.0	0.1	0.3	0.2	1.4	1.8
Average bitrate [kbps]	1219	1153	1012	819	463	418

Table 6

SCReAM

Users/cell	1	2	3	5	8	10
Video tail latency [ms]	111	124	174	182	209	268
IP packet tail latency [ms]	95	101	118	128	154	170
Average PLR [%]	0.0	0.0	0.1	0.4	0.1	0.1
Average bitrate [kbps]	1286	1123	856	663	451	362

Table 7

## 11.2.2. RTT 40ms 30km/h

GCC

Users/cell	1	2	3	5	8	10
Video tail latency [ms]	434	733	913	996	1047	1066
IP packet tail latency [ms]	523	757	902	987	1046	1062
Average PLR [%]	0.3	0.5	0.9	0.9	2.3	2.7
Average bitrate [kbps]	1154	939	753	546	288	241

Table 8

SCReAM

Users/cell	1	2	3	5	8	10
Video tail latency [ms]	144	174	232	263	276	357
IP packet tail latency [ms]	109	113	134	141	167	215
Average PLR [%]	0.0	0.0	0.1	0.1	0.1	0.3
Average bitrate [kbps]	848	698	572	456	302	240

Table 9

## 12. References

### 12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, December 2012.

### 12.2. Informative References

- [FACK] "Forward Acknowledgement: Refining TCP Congestion Control", 2006.
- [I-D.alvestrand-rmcat-congestion] Holmer, S., Cicco, L., Mascolo, S., and H. Alvestrand, "A Google Congestion Control Algorithm for Real-Time Communication", draft-alvestrand-rmcat-congestion-02 (work in progress), February 2014.

- [I-D.draft-sarker-rmcat-cellular-eval-test-cases]  
Sarker, Z., "Evaluation Test Cases for Interactive Real-Time Media over Cellular Networks",  
<<http://www.ietf.org/id/draft-sarker-rmcat-cellular-eval-test-cases-00.txt>>.
- [I-D.ietf-tcpm-newcwv]  
Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to support Rate-Limited Traffic", draft-ietf-tcpm-newcwv-06 (work in progress), March 2014.
- [QoS-3GPP]  
TS 23.203, 3GPP., "Policy and charging control architecture", June 2011, <[http://www.3gpp.org/ftp/specs/archive/23\\_series/23.203/23203-990.zip](http://www.3gpp.org/ftp/specs/archive/23_series/23.203/23203-990.zip)>.
- [TFWC]  
University College London, "Fairer TCP-Friendly Congestion Control Protocol for Multimedia Streaming", December 2007, <<http://www-dept.cs.ucl.ac.uk/staff/M.Handley/papers/tfwc-conext.pdf>>.

## Authors' Addresses

Ingemar Johansson  
Ericsson AB  
Laboratoriegraend 11  
Luleae 977 53  
Sweden

Phone: +46 730783289  
Email: [ingemar.s.johansson@ericsson.com](mailto:ingemar.s.johansson@ericsson.com)

Zaheduzzaman Sarker  
Ericsson AB  
Laboratoriegraend 11  
Luleae 977 53  
Sweden

Phone: +46 761153743  
Email: [zaheduzzaman.sarker@ericsson.com](mailto:zaheduzzaman.sarker@ericsson.com)

RMCAT WG  
Internet-Draft  
Intended status: Informational  
Expires: September 3, 2015

I. Johansson  
Z. Sarker  
Ericsson AB  
March 2, 2015

Self-Clocked Rate Adaptation for Multimedia  
draft-johansson-rmcat-scream-cc-05

Abstract

This memo describes a rate adaptation algorithm for conversational video services. The solution conforms to the packet conservation principle and uses a hybrid loss and delay based congestion control algorithm. The algorithm is evaluated over both simulated Internet bottleneck scenarios as well as in a LTE (Long Term Evolution) system simulator and is shown to achieve both low latency and high video throughput in these scenarios.

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 3, 2015.

Copyright Notice

Copyright (c) 2015 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 . . . . .	2
1.1. Wireless (LTE) access properties . . . . .	3
2. Terminology . . . . .	3
3. Overview of SCReAM Algorithm . . . . .	3
3.1. Congestion Control . . . . .	4
3.2. Transmission Scheduling . . . . .	5
3.3. Media Rate Control . . . . .	5
4. Detailed Description of SCReAM . . . . .	5
4.1. SCReAM Sender . . . . .	5
4.1.1. Constants and Parameter values . . . . .	7
4.1.2. Network congestion control . . . . .	11
4.1.2.1. Congestion window update . . . . .	12
4.1.2.2. Transmission scheduling . . . . .	15
4.1.3. Video rate control . . . . .	16
4.2. SCReAM Receiver . . . . .	19
5. Feedback Message . . . . .	20
6. Additional features . . . . .	21
6.1. Packet pacing . . . . .	21
6.2. Frame skipping . . . . .	21
6.3. Q-bit semantics (source quench) . . . . .	23
7. Discussion . . . . .	23
8. Conclusion . . . . .	24
9. Open issues . . . . .	24
10. Source code . . . . .	25
11. Acknowledgements . . . . .	25
12. IANA Considerations . . . . .	25
13. Security Considerations . . . . .	25
14. Change history . . . . .	25
15. References . . . . .	26
15.1. Normative References . . . . .	26
15.2. Informative References . . . . .	26
Authors' Addresses . . . . .	27

## 1. Introduction

Congestion in the internet is a reality and applications that are deployed in the internet must have congestion control schemes in place not only for the robustness of the service that it provides but also to ensure the function of the currently deployed internet. As the interactive realtime communication imposes a great deal of requirements on the transport, a robust, efficient rate adaptation for all access types is considered as an important part of interactive realtime communications as the transmission channel

bandwidth may vary over time. Wireless access such as LTE, which is an integral part of the current internet, increases the importance of rate adaptation as the channel bandwidth of a default LTE bearer [QoS-3GPP] can change considerably in a very short time frame. Thus a rate adaptation solution for interactive realtime media, such as WebRTC, must be both quick and be able to operate over a large span in available channel bandwidth. This memo describes a solution, named SCReAM, that is based on the self-clocking principle of TCP and uses techniques similar to what is used in a new delay based rate adaptation algorithm, LEDBAT [RFC6817]. Because neither TCP nor LEDBAT was designed for interactive realtime media, a few extra features are needed to make the concept work well within this context. This memo describes these extra features.

### 1.1. Wireless (LTE) access properties

[I-D.draft-sarker-rmcat-cellular-eval-test-cases] introduces the complications that can be observed in wireless environments. Wireless access such as LTE can typically not guarantee a given bandwidth, this is true especially for default bearers. The network throughput may vary considerably for instance in cases where the wireless terminal is moving around.

Unlike wireline bottlenecks with large statistical multiplexing it is not possible to try to maintain a given bitrate when congestion is detected with the hope that other flows will yield, this because there are generally few other flows competing for the same bottleneck. Each user gets its own variable throughput bottleneck, where the throughput depends on factors like channel quality, network load and historical throughput. The bottom line is, if the throughput drops, the sender has no other option than to reduce the bitrate. In addition, the grace time, i.e. allowed reaction time from the time that the congestion is detected until a reaction in terms of a rate reduction is effected, is generally very short, in the order of one RTT (Round Trip Time).

## 2. Terminology

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 RFC2119 [RFC2119]

## 3. Overview of SCReAM Algorithm

The core SCReAM algorithm has similarities to concepts like self-clocking used in TFWC [TFWC] and follows packet conservation principles. The packet conservation principle is described as an

important key-factor behind the protection of networks from congestion [FACK].

The packet conservation principle is realized by including an indication of the highest received sequence number in the feedback, see Section 5, from the receiver back to the sender, the sender keeps a list of transmitted packets and their respective sizes. This information is then used to determine how many bytes can be transmitted. A congestion window puts an upper limit on how many bytes can be in flight, i.e. transmitted but not yet acknowledged. The congestion window is determined in a way similar to LEDBAT [RFC6817]. This ensures that the e2e latency is kept low. The basic functionality is quite simple, there are however a few steps to take to make the concept work with conversational media. These will be briefly described in sections Section 3.1 to Section 3.3.

The rate adaptation solution constitutes three parts- congestion control, transmission scheduling and media rate adaptation. All these three parts reside at the sender side. The receiver side algorithm is very simple in comparison as it only generates acknowledgements to received RTP packets.

### 3.1. Congestion Control

The congestion control sets an upper limit on how much data can be in the network (bytes in flight); this limit is called CWND (congestion window) and is used in the transmission scheduling.

The SCReAM congestion control method, uses LEDBAT [RFC6817] to measure the OWD (one way delay). The SCReAM sender calculates the congestion window based on the feedback from SCReAM receiver. The congestion window is allowed to increase if the OWD is below a predefined target, otherwise the congestion window decreases. The delay target is typically set to 50-100ms. This ensures that the OWD is kept low on the average. The reaction to loss events is similar to that of loss based TCP, i.e. an instant reduction of CWND.

LEDBAT is designed with file transfers as main use case which means that the algorithm must be modified somewhat to work with rate-limited sources such as video. The modifications are

- o Congestion window validation techniques. These are similar in action as the method described in [I-D.ietf-tcpm-newcwnd].
- o Fast start for bitrate increase. It makes the video bitrate ramp-up within 5 to 10 seconds. The behavior is similar to TCP slowstart. The fast start is exited when congestion is detected. The fast start state can be resumed if the congestion level is

low, this to enable a reasonably quick rate increase in case link throughput increases.

- o Adaptive delay target. This helps the congestion control to compete with FTP traffic to some degree.

### 3.2. Transmission Scheduling

Transmission scheduling limits the output of data, given by the relation between the number of bytes in flight and the congestion window similar to TCP. Packet pacing is used to mitigate issues with coalescing that may cause increased jitter and/or packet loss in the media traffic.

### 3.3. Media Rate Control

The media rate control serves to adjust the media bitrate to ramp up quickly enough to get a fair share of the system resources when link throughput increases.

The reaction to reduced throughput must be prompt in order to avoid getting too much data queued up in the RTP packet queues. The media bitrate is decreased if the RTP queue size exceeds a threshold.

In cases where the sender frame queues increase rapidly such as the case of a RAT (Radio Access Type) handover it may be necessary to implement additional actions, such as discarding of encoded video frames or frame skipping in order to ensure that the RTP queues are drained quickly. Frame skipping means that the frame rate is temporarily reduced. Discarding of old video frames is a more efficient way to reduce media latency than frame skipping but it comes with a requirement to repair codec state, frame skipping is thus to prefer as a first remedy. Frame skipping is described as an optional to implement feature in this specification.

## 4. Detailed Description of SCReAM

### 4.1. SCReAM Sender

This section describes the sender side algorithm in more detail. It is split between the network congestion control and the video rate adaptation.

Figure 1 shows the functional overview of a SCReAM sender. The RTP application interaction with congestion control is described in [I-D.ietf-rmcat-app-interaction]. Here we use a more decomposed version of the implementation model in the sense that the RTP packets may be queued up in the sender, the transmission of these RTP packets

is controlled by a transmission scheduler. A SCReAM sender implements rate control and a queue for each media type or source, where RTP packets containing encoded media frames are temporarily stored for transmission, the figure shows the details for when two video sources (a.k.a streams) are used.

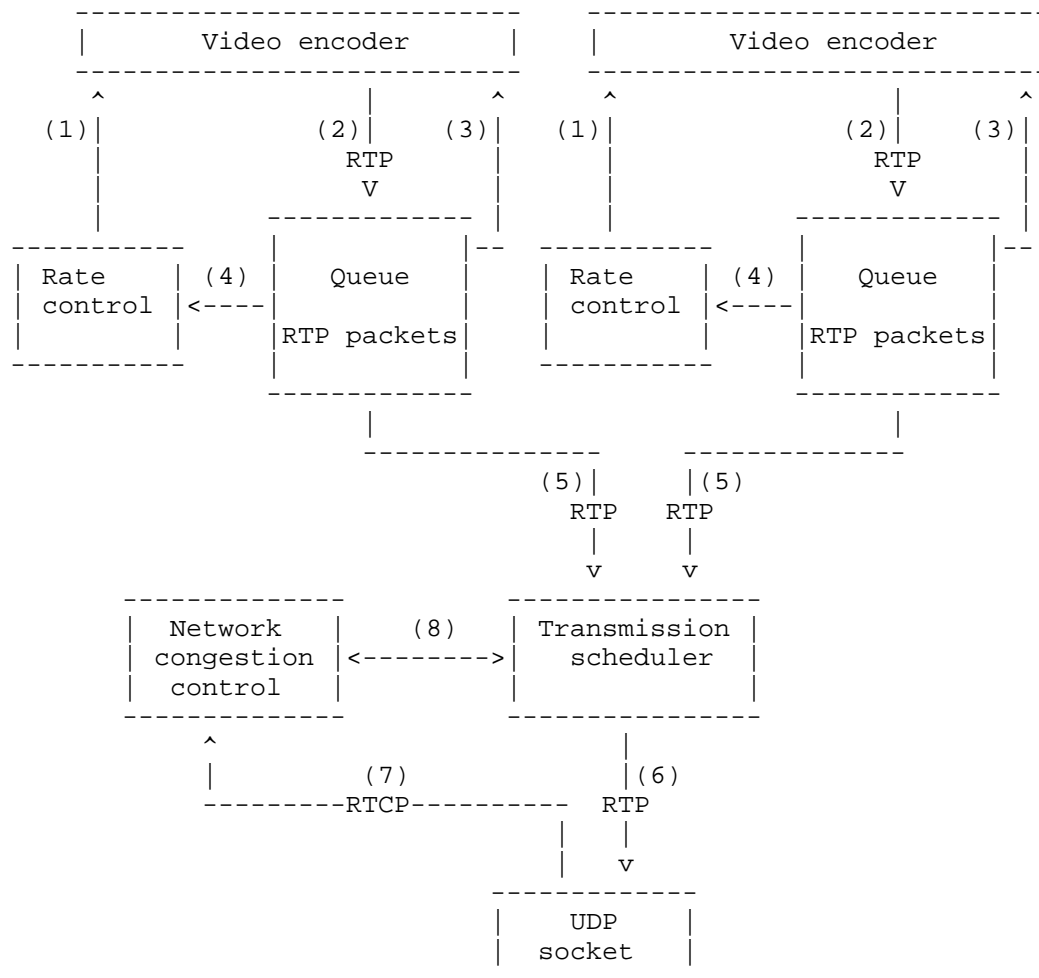


Figure 1: SCReAM sender functional view

Video frames are encoded and forwarded to the queue (2). The media rate adaptation adapts to the size of the RTP queue and controls the video bitrate (1). The RTP packets are picked from each queue based on some defined priority order or simply in a round robin fashion (5). A transmission scheduler takes care of the transmission of RTP

packets, to be written to the UDP socket (6). In the general case all media must go through the transmission scheduler and is allowed to be transmitted if the number of bytes in flight is less than the congestion window. Audio frames can however be allowed to be transmitted immediately as audio is typically low bitrate and thus contributes little to congestion, this is something that is left as an implementation choice. RTCP packets are received (7) and the information about bytes in flight and congestion window is exchanged between the network congestion control and the transmission scheduler (8).

#### 4.1.1. Constants and Parameter values

A set of constants are defined in Table 1, state variables are defined in Table 2. And finally, local variables are described in Table 3.

An init value [] indicates an empty array.

Constant	Explanation	Value
OWD_TARGET_LO	Min OWD target	0.1s
OWD_TARGET_HI	Max OWD target	0.4s
MAX_BYTES_IN_FLIGHT_HEAD_ROOM	Headroom for limitation of CWND	1.1
GAIN	Gain factor for congestion window adjustment	1.0
BETA	CWND scale factor due to loss event	0.6
BETA_R	Target rate scale factor due to loss event	0.8
BYTES_IN_FLIGHT_SLACK	Additional slack [%] to the congestion window	10%
RATE_ADJUST_INTERVAL	Interval between video bitrate adjustments	0.1s
FRAME_PERIOD	Video coder frame period [s]	
TARGET_BITRATE_MIN	Min target_bitrate [bps]	
TARGET_BITRATE_MAX	Max target_bitrate [bps]	
RAMP_UP_TIME	Timespan [s] from lowest to highest bitrate	10s
PRE_CONGESTION_GUARD	Guard factor against early congestion onset. A higher value gives less jitter possibly at the expense of a lower video bitrate.	0.0..0.2
TX_QUEUE_SIZE_FACTOR	Guard factor against RTP queue buildup	0.0..2.0

Table 1: Constants

Variable	Explanation	Init value
owd_target	OWD target	OWD_TARGET_LO
owd_fraction_avg	EWMA filtered owd_fraction	0.0
owd_fraction_hist	Vector of the last 20 owd_fraction	[]
owd_trend	OWD trend, indicates incipient congestion	0.0
owd_norm_hist	Vector of the last 100 owd_norm	[]
mss	Maximum segment size = Max RTP packet size [byte]	1000
min_cwnd	Minimum congestion window [byte]	2*MSS
in_fast_start	True if in fast start state	true
cwnd	Congestion window [byte]	min_cwnd
cwnd_i	Congestion window inflection point	1
bytes_newly_acked	The number of bytes that was acknowledged with the last received acknowledgement i.e. bytes acknowledged since the last CWND update [byte]. Reset after a CWND update	0
send_wnd	Upper limit of how many bytes that can be transmitted [byte]. Updated when CWND is updated and when RTP packet is transmitted	0
t_pace	Approximate estimate of inter- packet transmission	0.001

	interval [s], updated when RTP packet transmitted	
age_vec	A vector of the last 20 RTP packet queue delay samples	[]
frame_skip_intensity	Indicates the intensity of the frame skips	0.0
since_last_frame_skip	Number of video frames since the last skip	0
consecutive_frame_skips	Number of consecutive frame skips	0
target_bitrate	Video target bitrate [bps]	TARGET_BITRATE_MIN
target_bitrate_i	Video target bitrate inflection point i.e. the last known highest target_bitrate during fast start. Used to limit bitrate increase close to the last known congestion point	1
rate_transmit	Measured transmit bitrate [bps]	0.0
rate_acked	Measured throughput based on received acknowledgements [bps]	0.0
s_rtt	Smoothed RTT [s], computed similar to method depicted in [RFC6298]	0.0
rtp_queue_size	Size of RTP packets in queue [bits]	0
rtp_size	Size of the last transmitted RTP packets [byte]	0
frame_skip	Skip encoding of video frame if	false

	true	
+-----+	+-----+	+-----+

Table 2: State variables

Variable	Explanation
owd	OWD = One way delay with base delay subtracted [s]. This is an estimate of the network queueing delay.
owd_fraction	OWD as a fraction of the OWD target
owd_norm	OWD normalized to OWD_TARGET_LO
owd_norm_mean	Average OWD norm over the last 100 samples
owd_norm_mean_sh	Average OWD norm over the last 20 samples
owd_norm_var	OWD norm variance over the last 100 samples
off_target	Relation between OWD and OWD target
scl_i	A general scalefactor that is applied to the CWND or target_bitrate increase
x_cwnd	Additional increase of CWND, used when send_wnd is computed
pace_bitrate	The allowed RTP packet transmission rate, used in the computation of t_pace [bps]
age_avg	Average RTP queue delay [s]
increment	Allowed target_bitrate increase
current_rate	Max of rate_transmit and rate_acked

Table 3: Local temporary variables

#### 4.1.2. Network congestion control

This section explains the network congestion control, it contains two main functions

- o Computation of congestion window at the sender: Gives an upper limit to the number of bytes in flight i.e. how many bytes that have been transmitted but not yet acknowledged.
- o Transmission scheduling at the sender: RTP packets are transmitted if allowed by the relation between the number of bytes in flight and the congestion window. This is controlled by the send window.

Unlike TCP, SCReAM is not a byte oriented protocol, rather it is an RTP packet oriented protocol. Thus it keeps a list of transmitted RTP packets and their respective sending times (wall-clock time). The feedback indicates the highest received RTP sequence number and a

timestamp (wall-clock time) when it was received. In addition, an ACK list is included to make it possible to determine lost packets.

#### 4.1.2.1. Congestion window update

The congestion window is computed from the one way (extra) delay estimates (OWD) that are obtained from the send and received timestamp of the RTP packets. LEDBAT [RFC6817] explains the details of the computation of the OWD. An OWD sample is obtained for each received acknowledgement. No smoothing of the OWD samples occur, however some smoothing occurs anyway as the computation of the CWND is in itself a low pass filter function.

SCReAM uses the terminology "Bytes in flight (`bytes_in_flight`)" which is computed as the sum of the sizes of the RTP packets ranging from the RTP packet most recently transmitted down to but not including the acknowledged packet with the highest sequence number. As an example: If RTP packet was sequence number SN with transmitted and the last ACK indicated SN-5 as the highest received sequence number then bytes in flight is computed as the sum of the size of RTP packets with sequence number SN-4, SN-3, SN-2, SN-1 and SN.

CWND is updated differently depending on whether the congestion control is in fast start or not and if a loss event is detected. A Boolean variable `in_fast_start` indicates if the congestion is in fast start state.

A loss event indicates one or more lost RTP packets within an RTT. This is detected by means of inspection for holes in the sequence number space in the acknowledgements with some margin for possible packet reordering in the network. As an alternative, a timer for loss detection similar to TCP RACK may be used.

Below is described the actions when an acknowledgement from the receiver is received.

`bytes_newly_acked` is updated.

The OWD fraction and an average of it are computed as

$$\text{owd\_fraction} = \text{owd} / \text{owd\_target}$$
$$\text{owd\_fraction\_avg} = 0.9 * \text{owd\_fraction\_avg} + 0.1 * \text{owd\_fraction}$$

The OWD fraction is sampled every 50ms and the last 20 samples are stored in a vector (`owd_fraction_hist`). This vector is used in the computation of an OWD trend that gives a value between 0.0 and 1.0

depending on how close to congestion it is. The OWD trend is calculated as follows

Let  $R(\text{owd\_fraction\_hist}, K)$  be the autocorrelation function of  $\text{owd\_fraction\_hist}$  at lag  $K$ . The 1st order prediction coefficient is formulated as

$$a = R(\text{owd\_fraction\_hist}, 1) / R(\text{owd\_fraction\_hist}, 0)$$

The prediction coefficient  $a$  has positive values if OWD shows an increasing trend, thus an indication of congestion is obtained before the OWD target is reached. The prediction coefficient is further multiplied with  $\text{owd\_fraction\_avg}$  to reduce sensitivity to increasing OWD when OWD is very small. The OWD trend is thus computed as

$$\text{owd\_trend} = \max(0.0, \min(1.0, a * \text{owd\_fraction\_avg}))$$

The  $\text{owd\_trend}$  is utilized in the media rate control and to determine when to exit slow start.

An off target value is computed as

$$\text{off\_target} = (\text{owd\_target} - \text{owd}) / \text{owd\_target}$$

A temporal variable  $\text{scl\_i}$  is computed as

$$\text{scl\_i} = \max(0.2, \min(1.0, (\text{abs}(\text{cwnd} - \text{cwnd\_i}) / \text{cwnd\_i} * 4)^2))$$

$\text{scl\_i}$  is used to limit the CWND increase when close to the last known max value, before congestion was last detected.

The congestion window update depends on whether a loss event has occurred, and if the congestion control is if fast start or not.

---

On loss event:

If a loss event is detected then  $\text{in\_fast\_start}$  is set to false and CWND is updated according to

$$\text{cwnd\_i} = \text{cwnd}$$
$$\text{cwnd} = \max(\text{min\_cwnd}, \text{cwnd} * \text{BETA})$$

otherwise the CWND update continues

---

`in_fast_start = true:`

`in_fast_start` is set to false and `cwnd_i=cwnd` if `owd_trend >= 0.2` and otherwise CWND is updated according to

`cwnd = cwnd + bytes_newly_acked*scl_i`

---

`in_fast_start = false:`

Values of `off_target > 0.0` indicates that the congestion window can be increased. This is done according to the equations below.

`gain = GAIN*(1.0 + max(0.0, 1.0 - owd_trend/ 0.2))`

The equation above limits the gain when near congestion is detected

`gain *= scl_i`

This equation limits the gain when CWND is close to its last known max value

`cwnd += gain * off_target * bytes_newly_acked * mss / cwnd`

Values of `off_target <= 0.0` indicates congestion, CWND is then updated according to the equation

`cwnd += GAIN*off_target*bytes_newly_acked*mss/cwnd`

The equations above are very similar to what is specified in [RFC6817]. There are however a few differences.

- o [RFC6817] specifies a constant GAIN, this specification however limits the gain when CWND is increased dependent on near congestion state and the relation to the last known max CWND value.
  - o [RFC6817] specifies that the CWND increased is limited by an additional function controlled by a constant ALLOWED\_INCREASE. This additional limitation is removed in this specification.
- 

A number of final steps in the congestion window update procedure are outlined below

---

Resume fast start:

Fast start can be resumed in order to speed up the bitrate increase in case congestion abates. The condition to resume fast start (`in_fast_start = true`) is that `owd_trend` is less than 0.2 for 1.0 seconds or more.

---

Competing flows compensation, adjustment of `owd_target`:

Competing flows compensation is needed to avoid that flows congestion controlled by SCReAM are starved out by flows that are more aggressive in their nature. The `owd_target` is adjusted according to the `owd_norm_mean_sh` whenever `owd_norm_var` is below a given value. The condition to update `owd_target` is fulfilled if `owd_norm_var < 0.16` (indicating that the standard deviation is less than 0.4). `owd_target` is then update as:

```
owd_target = min(OWD_TARGET_HI, max(OWD_TARGET_LO, owd_norm_mean_sh*OWD_TARGET_LO*1.1))
```

---

Final CWND adjustment step:

The congestion window is limited by the maximum number of bytes in flight over the last 1.0 seconds according to

```
cwnd = min(cwnd, max_bytes_in_flight*MAX_BYTES_IN_FLIGHT_HEAD_ROOM)
```

This avoids possible over-estimation of the throughput after for example, idle periods.

Finally `cwnd` is set to ensure that it is at least `min_cwnd`

```
cwnd = max(cwnd, MIN_CWND)
```

#### 4.1.2.2. Transmission scheduling

The principle is to allow packet transmission of an RTP packet only if the number of bytes in flight is less than the congestion window. There are however two reasons why this strict rule will not work optimally:

- o Bitrate variations: The video frame size is always varying to a larger or smaller extent, a strict rule as the one given above will have the effect that the video bitrate have difficulties to increase as the congestion window puts a too hard restriction on the video frame size variation, this further can lead to occasional queuing of RTP packets in the RTP packet queue that will prevent bitrate increase because of the increased RTP queue size.
- o Reverse (feedback) path congestion: Especially in transport over buffer-bloated networks, the one way delay in the reverse direction may jump due to congestion. The effect of this is that the acknowledgements are delayed with the result that the self-clocking is temporarily halted, even though the forward path is not congested.

Packets are transmitted at a pace given by the send window, computed below

The send window is computed differently depending on OWD and its relation to the OWD target.

- o If  $owd > owd\_target$ :  
The send window is computed as  
 $send\_wnd = cwnd - bytes\_in\_flight$   
This enforces a strict rule that helps to prevent further queue buildup.
- o If  $owd \leq owd\_target$ :  
A helper variable  
 $x\_cwnd = 1.0 + BYTES\_IN\_FLIGHT\_SLACK * \max(0.0, \min(1.0, 1.0 - owd\_trend / 0.5)) / 100.0$   
is computed. The send window is computed as  
 $send\_wnd = \max(cwnd * x\_cwnd, cwnd + mss) - bytes\_in\_flight$   
This gives a slack that reduces as congestion increases,  $BYTES\_IN\_FLIGHT\_SLACK$  is a maximum allowed slack in percent. A large value increases the robustness to bitrate variations in the source and congested feedback channel issues. The possible drawback is increased delay or packet loss when forward path congestion occur.

#### 4.1.3. Video rate control

The video rate control is operated based on the size of the RTP packet send queue and observed loss events. In addition,  $owd\_trend$  is also considered in the rate control, this to reduce the amount of induced network jitter.

A variable `target_bitrate` is adjusted depending on the congestion state. The target bitrate can vary between a minimum value (`target_bitrate_min`) and a maximum value (`target_bitrate_max`).

For the overall bitrate adjustment, two network throughput estimates are computed :

- o `rate_transmit`: The measured transmit bitrate
- o `rate_acked`: The ACKed bitrate, i.e. the volume of ACKed bits per time unit.

Both estimates are updated every 200ms.

The current throughput `current_rate` is computed as the maximum value of `rate_transmit` and `rate_acked`. The rationale behind the use of `rate_acked` in addition to `rate_transmit` is that `rate_transmit` is affected also by the amount of data that is available to transmit, thus a lack of data to transmit can be seen as reduced throughput that may itself cause an unnecessary rate reduction. To overcome this shortcoming; `rate_acked` is used as well. This gives a more stable throughput estimate.

The bitrate is updated at regular intervals, given by `RATE_ADJUST_INTERVAL` and differently depending the fast start state

The rate change behavior depends on whether a loss event has occurred, and if the congestion control is if fast start or not.

---

On loss event:

First of all the `target_bitrate` is updated if a new loss event was indicated and the rate change procedure is exited.

```
target_bitrate_i = target_bitrate
```

```
target_bitrate = max(BETA_R* target_bitrate, TARGET_BITRATE_MIN)
```

If no loss event was indicated then the rate change procedure continues.

---

`in_fast_start = true:`

An allowed increment is computed based on the congestion level and the relation to `target_bitrate_i`

`scl_i = (target_bitrate - target_bitrate_i) / target_bitrate_i`

`increment = TARGET_BITRATE_MAX * RATE_ADJUST_INTERVAL / RAMP_UP_TIME * (1.0 - min(1.0, owd_trend / 0.1))`

`increment *= max(0.2, min(1.0, (scl_i * 4) ^ 2))`

`target_bitrate += increment`

`target_bitrate` is reduced further if congestion is detected.

`target_bitrate *= (1.0 - PRE_CONGESTION_GUARD * owd_trend)`

`target_bitrate = min(TARGET_BITRATE_MAX, max(TARGET_BITRATE_MIN, target_bitrate))`

---

`in_fast_start = false:`

`target_bitrate_i` is updated to the current value of `target_bitrate` if `in_fast_start` was true the last time the bitrate was updated.

A pre-congestion indicator is computed as

`pre_congestion = min(1.0, max(0.0, owd_fraction_avg - 0.3) / 0.7)`

`pre_congestion += owd_trend`

The target bitrate is computed as

`target_bitrate = current_rate * (1.0 - PRE_CONGESTION_GUARD * pre_congestion) - TX_QUEUE_SIZE_FACTOR * rtp_queue_size`

`target_bitrate = min(TARGET_BITRATE_MAX, max(TARGET_BITRATE_MIN, target_bitrate))`

#### 4.2. SCReAM Receiver

The SCReAM receiver is very simple in its implementation. The task is to feedback acknowledgements of received packets. For that purpose a set of state variables are needed, these are explained in Table 4.

One set of state variables are maintained per stream.

Variable	Explanation	Init value
rx_timestamp	The wall clock timestamp when the latest RTP packet was received	0
highest_rtp_sequence_number	The highest received sequence number	0
ack_vector	A 16 bit vector that indicates received RTP packets with a sequence number lower than highest_rtp_sequence_number	0
n_loss	An 8 bit counter for the number of lost RTP packets, separate counters are maintained for each SSRC	0
n_ECN	An 8 bit counter for the number of ECN-CE marked RTP packets, separate counters are maintained for each SSRC	0
pending_feedback	Indicates that an RTP packet was received and that an RTCP packet can be generated when RTCP timing rules permit	false
last_transmit_t	Last time an RTCP packet was transmitted, this is used to ensure that RTCP feedback is generated fairly for all streams.	-1.0

Table 4: State variables

Upon reception of an RTP packet, the state variables in Table 4 should be updated and the RTCP processing function should be

notified. An RTCP packet is later generated based on the state variables, how often this is done depends on the RTCP bandwidth.

## 5. Feedback Message

The feedback is over RTCP [RFC3550] and is based on [RFC4585]. It is implemented as a transport layer feedback message (RTPFB), see proposed example in Figure 2. The feedback control information part (FCI) consists of the following elements.

- o Highest received RTP sequence number : The highest received RTP sequence number for the given SSRC
- o n\_lost : Accumulated number of lost RTP packets for the given SSRC
- o Timestamp: A timestamp value indicating when the last packet was received which makes it possible to compute the one way (extra) delay (OWD).
- o n\_ECN : Accumulated number of ECN-CE marked RTP packets for the given SSRC
- o Source quench bit (Q): Makes it possible to request the sender to reduce its congestion window. This is useful if WebRTC media is received from many hosts and it becomes necessary to balance the bitrates between the streams.

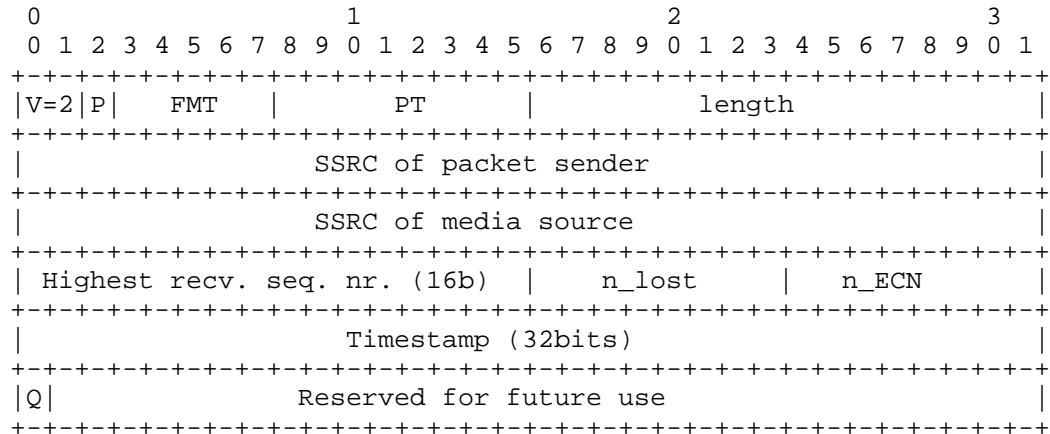


Figure 2: Transport layer feedback message

To make the feedback as frequent as possible, the feedback packets are transmitted as reduced size RTCP according to [RFC5506].

The timestamp clock time is recommended to be set to a fixed value such as 1000Hz, defined in this specification. The `n_lost` and `n_ECN` makes it possible to take necessary actions on the detection of lost and ECN marked packets.

Section 4 describes the main algorithm details and how the feedback is used.

## 6. Additional features

This section describes additional features. They are not required for the basic functionality of SCReAM but can improve performance in certain scenarios and topologies.

### 6.1. Packet pacing

Packet pacing is used in order to mitigate coalescing i.e. that packets are transmitted in bursts.

Packet pacing is enforced when `owd_fraction_avg` is greater than 0.1. The time interval between consecutive packet transmissions is then enforced to equal or higher than `t_pace` where `t_pace` is given by the equations below.

$$\text{pace\_bitrate} = \max(50000, \text{cwnd} * 8 / \text{s\_rtt})$$
$$\text{t\_pace} = \text{rtp\_size} * 8 / \text{pace\_bitrate}$$

`rtp_size` is the size of the last transmitted RTP packet

### 6.2. Frame skipping

Frame skipping is a feature that makes it possible to reduce the size of the RTP queue in the cases that e.g. the channel throughput drops dramatically or even goes below the lowest possible video coder rate. Frame skipping is optional to implement as it can sometimes be difficult to realize e.g. due to lack of API function to support this.

Frame skipping is controlled by a flag `frame_skip` which, if set to 1 dictates that the video coder should skip the next video frame. The frame skipping intensity at the current time instant is computed according to the steps below

The queuing delay is sampled every frame period and the last 20 samples are stored in a vector `age_vec`

An average queuing delay is computed as a weighted sum over the samples in `age_vec`. `age_avg` at the current time instant is computed as

$$\text{age\_avg}(n) = \text{SUM } \text{age\_vec}(n-k) * w(k) \quad k = [0..20[$$

$w(n)$  are weight factors arranged to give the most recent samples a higher weight.

The change in `age_avg` is computed as

$$\text{age\_d} = \text{age\_avg}(n) - \text{age\_avg}(n-1)$$

The frame skipping intensity at the current time instant  $n$  is computed as

- o If `age_d > 0` and `age_avg > 2*FRAME_PERIOD`:  
    `frame_skip_intensity = min(1.0, (age_vec(n)-2*FRAME_PERIOD)/(4*FRAME_PERIOD))`
- o Otherwise frame skip intensity is set to zero

The `skip_frame` flag is set depending on three variables

- o `frame_skip_intensity`
- o `since_last_frame_skip`, i.e the number of consecutive frames without frame skipping
- o `consecutive_frame_skips`, i.e the number of consecutive frame skips

The flag `skip_frame` is set to 1 if any of the conditions below is met, otherwise it is set to 0.

- o `age_vec(n) > 0.2 && consecutive_frame_skips < 5`
- o `frame_skip_intensity < 0.5 && since_last_frame_skip >= 1.0 / frame_skip_intensity`
- o `frame_skip_intensity >= 0.5 && consecutive_frame_skips < (frame_skip_intensity - 0.5) * 10`

The arrangement makes sure that no more than 4 frames are skipped in sequence, the rationale is to ensure that the input to the video encoder does not change too much, something that may give poor prediction gain.

### 6.3. Q-bit semantics (source quench)

The Q bit in the feedback is set by a receiver to signal that the sender should reduce the bitrate. The sender will in response to this reduce the congestion window with the consequence that the video bitrate decreases. A typical use case for source quench is when a receiver receives streams from sources located at different hosts and they all share a common bottleneck, typically it is difficult to apply any rate distribution signaling between the sending hosts. The solution is then that the receiver sets the Q bit in the feedback to the sender that should reduce its rate, if the streams share a common bottleneck then the released bandwidth due to the reduction of the congestion window for the flow that had the Q bit set in the feedback will be grabbed by the other flows that did not have the Q bit set. This is ensured by the opportunistic behavior of SCReAM's congestion control. The source quench will have no or little effect if the flows do not share the same bottleneck.

The reduction in congestion window is proportional to the amount of SCReAM RTCP feedback with the Q bit set, the below steps outline how the sender should react to RTCP feedback with the Q bit set. The reduction is done once per RTT. Let :

- o  $n$  = Number of received RTCP feedback messages in one RTT
- o  $n_q$  = Number of received RTCP feedback messages in one RTT, with Q bit set.

The new congestion window is then expressed as:

$$cwnd = \max(\text{MIN\_CWND}, cwnd * (1.0 - 0.5 * n_q / n))$$

Note that CWND is adjusted at most once per RTT. Furthermore The CWND increase should be inhibited for one RTT if CWND has been decreased as a result of Q bits set in the feedback.

The required intensity of the Q-bit set in the feedback in order to achieve a given rate distribution depends on many factors such as RTT, video source material etc. The receiver thus need to monitor the change in the received video bitrate on the different streams and adjust the intensity of the Q-bit accordingly.

## 7. Discussion

This section covers a few open discussion points

- o RTCP feedback overhead: SCReAM benefits from a relatively frequent feedback. Experiments have shown that a feedback rate roughly

equal to the frame rate gives a stable self-clocking and robustness against loss of feedback. With a maximum bitrate of 1500kbps the RTCP feedback overhead is in the range 10-15kbps with reduced size RTCP, including IP and UDP framing, in other words the RTCP overhead is quite modest and should not pose a problem in the general case. Other solutions may be required in highly asymmetrical link capacity cases. Worth notice is that SCReAM can work with as low feedback rates as once every 200ms, this however comes with a higher sensitivity to loss of feedback and also a potential reduction in throughput.

- o AVPF mode: The RTCP feedback is based on AVPF regular mode. The SCReAM feedback is transmitted as reduced size RTCP so save overhead, it is however required to transmit full compound RTCP at regular intervals, this interval can be controlled by trr-int depicted in [RFC4585].
- o BETA, CWND scale factor due to loss: The BETA value is recommended to be higher than 0.5. The reason behind this is that congestion control for multimedia has to deal with a source that is rate limited. A file transfer has "unlimited" source bitrate in comparison. The outcome is that SCReAM must be a little more aggressive than a file transfer in order to not be out competed.

## 8. Conclusion

This memo describes a congestion control algorithm for RMCAT that it is particularly good at handling the quickly changing condition in wireless network such as LTE. The solution conforms to the packet conservation principle and leverages on novel congestion control algorithms and recent TCP research, together with media bitrate determined by sender queuing delay and given delay thresholds. The solution has shown potential to meet the goals of high link utilization and prompt reaction to congestion. The solution is realized with a new RFC4585 transport layer feedback message.

## 9. Open issues

A list of open issues.

- o Describe how clock drift compensation is done
- o Describe how FEC overhead is accounted for in target\_bitrate computation
- o Investigate the impact of more sparse RTCP feedback, for instance once per RTT

## 10. Source code

Source code for SCReAM is available in two formats :

- o C++ code, that is apt for experimentation. The code maintained as Visual Studio project. This code can possibly be included in simulators such as NS3. Available at <https://github.com/EricssonResearch/scream>
- o OpenWebRTC implementation : Work in progress, see <http://www.openwebrtc.io/> for information about the OpenWebRTC project

## 11. Acknowledgements

We would like to thank the following persons for their comments, questions and support during the work that led to this memo: Markus Andersson, Bo Burman, Tomas Frankkila, Frederic Gabin, Laurits Hamm, Hans Hannu, Nikolas Hermanns, Stefan Haekansson, Erlendur Karlsson, Daniel Lindstroem, Mats Nordberg, Jonathan Samuelsson, Rickard Sjoeborg, Robert Swain, Magnus Westerlund, Stefan Aelund.

## 12. IANA Considerations

A new RFC4585 transport layer feedback message needs to be standardized.

## 13. Security Considerations

The feedback can be vulnerable to attacks similar to those that can affect TCP. It is therefore recommended that the RTCP feedback is at least integrity protected.

## 14. Change history

A list of changes:

- o -04 to -05 : ACK vector is replaced by a loss counter, PT is removed from feedback, references to source code added
- o -03 to -04 : Extensive changes due to review comments, code somewhat modified, frame skipping made optional
- o -02 to -03 : Added algorithm description with equations, removed pseudo code and simulation results
- o -01 to -02 : Updated GCC simulation results

- o -00 to -01 : Fixed a few bugs in example code

## 15. References

### 15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, June 2011.
- [RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, December 2012.

### 15.2. Informative References

- [FACK] "Forward Acknowledgement: Refining TCP Congestion Control", 2006.
- [I-D.draft-sarker-rmcat-cellular-eval-test-cases] Sarker, Z., "Evaluation Test Cases for Interactive Real-Time Media over Cellular Networks", <<http://www.ietf.org/id/draft-sarker-rmcat-cellular-eval-test-cases-00.txt>>.
- [I-D.ietf-rmcat-app-interaction] Zanaty, M., Singh, V., Nandakumar, S., and Z. Sarker, "RTP Application Interaction with Congestion Control", draft-ietf-rmcat-app-interaction-01 (work in progress), October 2014.

## [I-D.ietf-tcpm-newcwv]

Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to support Rate-Limited Traffic", draft-ietf-tcpm-newcwv-08 (work in progress), February 2015.

## [QoS-3GPP]

TS 23.203, 3GPP., "Policy and charging control architecture", June 2011, <[http://www.3gpp.org/ftp/specs/archive/23\\_series/23.203/23203-990.zip](http://www.3gpp.org/ftp/specs/archive/23_series/23.203/23203-990.zip)>.

## [TFWC]

University College London, "Fairer TCP-Friendly Congestion Control Protocol for Multimedia Streaming", December 2007, <<http://www-dept.cs.ucl.ac.uk/staff/M.Handley/papers/tfwc-conext.pdf>>.

## Authors' Addresses

Ingemar Johansson  
Ericsson AB  
Laboratoriegraend 11  
Luleae 977 53  
Sweden

Phone: +46 730783289  
Email: [ingemar.s.johansson@ericsson.com](mailto:ingemar.s.johansson@ericsson.com)

Zaheduzzaman Sarker  
Ericsson AB  
Laboratoriegraend 11  
Luleae 977 53  
Sweden

Phone: +46 761153743  
Email: [zaheduzzaman.sarker@ericsson.com](mailto:zaheduzzaman.sarker@ericsson.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 03, 2015

C. S. Perkins  
University of Glasgow  
July 02, 2014

Using RTP Control Protocol (RTCP) Feedback for Unicast Multimedia  
Congestion Control  
draft-perkins-rmcat-rtp-cc-feedback-01

Abstract

This memo discusses the types of congestion control feedback that it is possible to send using the RTP Control Protocol (RTCP), and their suitability of use in implementing congestion control for unicast multimedia applications.

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 03, 2015.

Copyright Notice

Copyright (c) 2014 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 . . . . .	2
2. Possible Models for RTCP Feedback . . . . .	2
3. What Feedback is Achievable With RTCP? . . . . .	4
3.1. Per-packet Feedback . . . . .	4
3.2. Per-frame Feedback . . . . .	4
3.3. Per-RTT Feedback . . . . .	6
4. Discussion and Conclusions . . . . .	6
5. Security Considerations . . . . .	7
6. IANA Considerations . . . . .	7
7. Acknowledgements . . . . .	7
8. Informative References . . . . .	7
Author's Address . . . . .	8

## 1. Introduction

The coming deployment of WebRTC systems raises the prospect that high quality video conferencing will see extremely wide use. To ensure the stability of the network in the face of this use, WebRTC systems will need to use some form of congestion control for their RTP-based media traffic. To develop such congestion control, it is necessary to understand the sort of congestion feedback that can be provided within the framework of RTP [RFC3550] and the RTP Control Protocol (RTCP). It then becomes possible to determine if this is sufficient for congestion control, or if some form of RTP extension is needed.

This memo considers the congestion feedback that can be sent using RTCP under the RTP/SAVPF profile [RFC5124] (the secure version of the RTP/AVPF profile [RFC4585]). This profile was chosen as it forms the basis for media transport in WebRTC [I-D.ietf-rtcweb-rtp-usage] systems. Nothing in this memo is specific to the secure version of the profile, or to WebRTC, however.

## 2. Possible Models for RTCP Feedback

Several questions need to be answered when providing RTCP reception quality feedback for congestion control purposes. These include:

- o How often is feedback needed?
- o How much overhead is acceptable?
- o How much, and what, data does each report contain?

The key question is how often does the receiver need to send feedback on the reception quality it is experiencing, and hence the congestion

state of the network? Traditional congestion control protocols, such as TCP, send acknowledgements with every packet (or, at least, every couple of packets). That is straight-forward and low overhead when traffic is bidirectional and acknowledgements can be piggybacked onto return path data packets. It can also be acceptable, and can have reasonable overhead, to send separate acknowledgement packets when those packets are much smaller than data packets. It becomes a problem, however, when there is no return traffic on which to piggyback acknowledgements, and when acknowledgements are similar in size to data packets; this can be the case for some forms of media traffic, especially for voice over IP (VoIP) flows, but less so for video.

When considering multimedia traffic, it might make sense to consider less frequent feedback. For example, it might be possible to send a feedback packet once per video frame, or once per network round trip time (RTT). This could still give sufficiently frequent feedback for the congestion control loop to be stable and responsive while keeping the overhead reasonable when the feedback cannot be piggybacked onto returning data. In this case, it is important to note that RTCP can send much more detailed feedback than simple acknowledgements. For example, if it were useful, it could be possible to use an RTCP extended report (XR) packet [RFC3611] to send feedback once per RTT comprising a bitmap of lost and received packets, with reception times, over that RTT. As long as feedback is sent frequently enough that the control loop is stable, and the sender is kept informed when data leaves the network (to provide an equivalent to ACK clocking in TCP), it is not necessary to report on every packet at the instant it is received (indeed, it is unlikely that a video codec can react instantly to a rate change anyway, and there is little point in providing feedback more often than the codec can adapt).

The amount of overhead due to congestion control feedback that is considered acceptable has to be determined. RTCP data is sent in separate packets to RTP data, and this has some cost in terms of additional header overhead compared to protocols that piggyback feedback on return path data packets. The RTP standards have long said that a 5% overhead for RTCP traffic generally acceptable, while providing the ability to change this fraction. Is this still the case for congestion control feedback? Or is there a desire to either see more responsive feedback and congestion control, possibility with a higher overhead, or is lower overhead wanted, accepting that this might reduce responsiveness of the congestion control algorithm?

Finally, the details of how much, and what, data is to be sent in each report will affect the frequency and/or overhead of feedback. There is a fundamental trade-off that the more frequently feedback packets are sent, the less data can be included in each packet to

keep the overhead constant. Does the congestion control need high rate but simple feedback (e.g., like TCP acknowledgements), or is it acceptable to send more complex feedback less often?

### 3. What Feedback is Achievable With RTCP?

#### 3.1. Per-packet Feedback

RTCP packets are sent as separate packets to RTP media data, and the protocol includes no mechanism for piggybacking an RTCP packet onto an RTP data packet. In addition, the RTCP timing rules are based on the size of the RTP session, the number of active senders, the RTCP packet size, and the configured RTCP bandwidth fraction, with randomisation to prevent synchronisation of reports; accordingly the RTCP packet transmission times are extremely unlikely to line up with RTP packet transmission times. As a result, RTCP cannot be used to send per-packet feedback in its current form.

All of these issues with using RTCP for per-packet feedback could be resolved in an update to the RTP protocol, of course. Such an update could change the RTCP timing rules, and might define a shim layer to allow multiplexing of RTP and RTCP into a single packet, or to extend the RTP header to piggyback feedback data. This sort of change would be a large, and almost certainly backwards incompatible, extension to the RTP protocol, and is unlikely to be completed quickly, but could be done if there was a need.

#### 3.2. Per-frame Feedback

Consider one of the simplest scenarios for WebRTC: a point to point video call between two end systems. There will be four RTP flows in this scenario, two audio and two video, with all four flows being active for essentially all the time (the audio flows will likely use voice activity detection and comfort noise to reduce the packet rate during silent periods, and does not cause the transmissions to stop).

Assume all four flows are sent in a single RTP session, each using a separate SSRC. Further, assume each SSRC sends RTCP reports for all other SSRCs in the session (i.e., the optimisations in [I-D.ietf-avtcore-rtp-multi-stream-optimisation] are not used, giving the worst case for the RTCP overhead). When all members are senders like this, the RTCP timing rules in Sections 6.2 and 6.3 of [RFC3550] and [RFC4585] reduce to:

$$\text{rtcp\_interval} = \text{avg\_rtcp\_size} * n / \text{rtcp\_bw}$$

where  $n$  is the number of members in the session, the `avg_rtcp_size` is measured in octets, and the `rtcp_bw` is the bandwidth available for RTCP, measured in octets per second (this will typically be 5% of the session bandwidth).

The average RTCP size will depend on the amount of feedback that is sent in each RTCP packet, on the number of members in the session, and on the size of source description (RTCP SDES) information sent. As a baseline, each RTCP packet will be a compound RTCP packet that contains an RTCP SR and an RTCP SDES packet. In the scenario above, each RTCP SR packet will contain three report blocks, once for each of the other RTP SSRCs sending data, for a total of 100 octets (this is 8 octets header, 20 octets sender info, and  $3 * 24$  octets report blocks). The RTCP SDES packet will comprise a header (4 octets), an originating SSRC (4 octets), a CNAME chunk, and padding. If the CNAME follows [RFC7022] and [I-D.ietf-rtcweb-rtp-usage] it will be 19 octets in size, and require 1 octet of padding. The resulting compound RTCP packet will be 128 octets in size. If sent in UDP/IPv4 with no IP options and using Secure RTP, which adds 20 (IPv4) + 8 (UDP) + 14 (SRTP with 80 bit Authentication tag), the `avg_rtcp_size` will therefore be 170 octets, including the header overhead. The value  $n$  in this scenario is 4, and the `rtcp_bw` is assumed to be 5% of the session bandwidth.

If it is desired to send RTCP feedback packets on average 30 times per second, to correspond to one RTCP report every frame for 30fps video, one can invert the above `rtcp_interval` calculation to get an `rtcp_bw` that gives an interval of 1/30th of a second or lower. This corresponds to an `rtcp_bw` of 20400 octets per second (since  $1/30 = 170 * 4 / 20400$ ). This is 163200 bits per second, which if 5% of the session bandwidth, gives a session bandwidth of approximately 3.3Mbps (i.e., 3.3Mbps media rate, plus an additional 5% for RTCP, to give a total data rate of approximately 3.4Mbps). That is, RTCP can report on every frame of video provided the session bandwidth is 3.3Mbps or larger, when every SSRC sends a report for every video frame (due to randomisation inherent in the RTCP timing rules, the actual RTCP transmission intervals will be within the range [0.0135, 0.0406]s, but will maintain an average RTCP transmission interval of 0.033s). This is not out of line with the expected session bandwidth for this type of application, suggesting the RTCP feedback can be used to provide per-frame congestion control feedback for WebRTC-style applications.

Note: To achieve the RTCP transmission intervals above the RTP/SAVPF profile with `T_rr_interval=0` is used, since even when using the reduced minimal transmission interval, the RTP/SAVPF profile would only allow sending RTCP at most every 0.11s (every third frame of video). Using RTP/SAVPF with `T_rr_interval=0` however is

capable of fully utilizing the configured 5% RTCP bandwidth fraction.

If additional feedback beyond the standard report block is needed, the session bandwidth needed will increase slightly. For example, with an additional 20 octets data being reported in each RTCP packet, the session bandwidth needed increases to 3.5Mbps for every SSRC to be able to report on every frame.

The above calculations highlight the baseline feasibility of RTCP congestion control feedback, but might not be the most appropriate usage of the RTCP bandwidth of all applications. Depending on needs, a less frequent usage of regular RTCP compound packets, controlled by `T_rr_interval` combined with using the reduced size RTCP packets, can achieve more frequent and useful reporting. Also the optimisations defined in [I-D.ietf-avtcore-rtp-multi-stream-optimisation] will reduce the amount of bandwidth consumed for reporting when each endpoint has multiple SSRCs.

It might also seem unnecessary to assign the same fraction of the RTCP bandwidth to reporting on the audio and video, since video is much higher rate, and so is presumably more likely to cause congestion. Sending audio and video in separate RTCP sessions with their own RTCP bandwidth fraction would give essentially double the RTCP bandwidth for each video source, since RTCP bandwidth fraction would be shared between two reporting SSRCs, rather than between the four reporting SSRCs in the single session case. This would hence reduce the session bandwidth needed to allow reports on every frame. Extensions to split RTCP bandwidth unequally between participants in a single session could be defined to allow this to work with a single RTP session on a single UDP port, or two standard RTP sessions could be run on a single port, using a demultiplexing shim. RTCP already allows for different bandwidth fractions between senders and receivers, so this is a relatively small change to the protocol.

### 3.3. Per-RTT Feedback

The arguments made in Section 3.2 apply to this case as well. The network RTT will usually be larger than the media framing interval, so sending feedback per RTT is less of a load on RTCP than sending feedback per frame.

## 4. Discussion and Conclusions

RTCP as it is currently specified cannot be used to send per-packet congestion feedback. RTCP can, however, be used to send congestion feedback on each frame of video sent, provided the session bandwidth exceeds a couple of megabits per second (the exact rate depending on

the number of session participants, the RTCP bandwidth fraction, and whether audio and video are sent in one or two RTP sessions). RTCP can likely also be used to send feedback on a per-RTT basis, provided the RTT is not too low.

If it is desired to use RTCP in something close to its current form for congestion feedback in WebRTC, the multimedia congestion control algorithm needs to be designed to work with feedback sent roughly each frame or each RTT, rather than per packet, since that fits within the limitations of RTCP. That feedback can be a little more complex than just an acknowledgement, provided care is taken to consider the impact of the extra feedback on the overhead, possibly allowing for a degree of semantic feedback, meaningful to the codec layer as well as the congestion control algorithm.

Further study of the scenarios of interest is needed, to ensure that the analysis presented is applicable to other media topologies, and to sessions with different data rates and sizes of membership.

## 5. Security Considerations

The security considerations of [RFC3550], [RFC4585], and [RFC5124] apply.

## 6. IANA Considerations

There are no actions for IANA.

## 7. Acknowledgements

Thanks to Magnus Westerlund for his feedback on Section 3.2.

## 8. Informative References

[I-D.ietf-avtcore-rtp-multi-stream-optimisation]  
Lennox, J., Westerlund, M., Wu, W., and C. Perkins,  
"Sending Multiple Media Streams in a Single RTP Session:  
Grouping RTCP Reception Statistics and Other Feedback",  
draft-ietf-avtcore-rtp-multi-stream-optimisation-02 (work  
in progress), February 2014.

[I-D.ietf-rtcweb-rtp-usage]  
Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time  
Communication (WebRTC): Media Transport and Use of RTP",  
draft-ietf-rtcweb-rtp-usage-15 (work in progress), May  
2014.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, September 2013.

Author's Address

Colin Perkins  
University of Glasgow  
School of Computing Science  
Glasgow G12 8QQ  
United Kingdom

Email: [csp@csp Perkins.org](mailto:csp@csp Perkins.org)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: December 19, 2014

Z. Sarker  
Ericsson AB  
V. Singh  
Aalto University  
X. Zhu  
M. Ramalho  
Cisco Systems  
June 17, 2014

Test Cases for Evaluating RMCAT Proposals  
draft-sarker-rmcat-eval-test-01

Abstract

The Real-time Transport Protocol (RTP) is used to transmit media in multimedia telephony applications, these applications are typically required to implement congestion control. The RMCAT working group is currently working on candidate algorithms for such interactive real-time multimedia applications. This document describes the test cases to be used in the performance evaluation of those candidate algorithms.

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 December 19, 2014.

Copyright Notice

Copyright (c) 2014 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. Terminology . . . . .	3
3. Structure of Test cases . . . . .	3
4. Recommended Evaluation Settings . . . . .	7
4.1. Evaluation metrics . . . . .	8
4.2. Path characteristics . . . . .	8
4.3. Media source . . . . .	9
5. Basic Test Cases . . . . .	10
5.1. Variable Available Capacity with Single RMCAT flow . . . . .	10
5.2. Variable Available Capacity with Multiple RMCAT flows . . . . .	12
5.3. Congested Feedback Link with Bi-directional RMCAT flows . . . . .	14
5.4. Competing Flows with Same RMCAT Algorithm . . . . .	16
5.5. Round Trip Time Fairness . . . . .	18
5.6. RMCAT Flow competing with a long TCP Flow . . . . .	20
5.7. RMCAT Flow competing with short TCP Flows . . . . .	22
5.8. Media Pause and Resume . . . . .	24
6. Other potential test cases . . . . .	26
6.1. Explicit Congestion Notification Usage . . . . .	26
6.2. Multiple Bottlenecks . . . . .	26
7. Wireless Access Links . . . . .	28
7.1. Cellular Network Specific Test Cases . . . . .	28
7.2. Wi-Fi Network Specific Test Cases . . . . .	28
8. Security Considerations . . . . .	29
9. IANA Considerations . . . . .	29
10. Acknowledgements . . . . .	29
11. References . . . . .	29
11.1. Normative References . . . . .	29
11.2. Informative References . . . . .	30
Appendix A. List of Network Parameters . . . . .	31
A.1. One-way Propagation Delay . . . . .	31
A.2. End-to-end Loss . . . . .	31
A.3. DropTail Router Queue Length . . . . .	32
Appendix B. Models . . . . .	32
B.1. Jitter models . . . . .	32
B.2. Loss generation model . . . . .	34
B.3. TCP traffic model . . . . .	35
Authors' Addresses . . . . .	35

## 1. Introduction

This memo describes a set of test cases for evaluating candidate RMCAT congestion control algorithm proposals, it is based on the guidelines enumerated in [I-D.ietf-rmcat-eval-criteria] and the requirements discussed in [I-D.ietf-rmcat-cc-requirements]. The test cases cover basic usage scenarios and are described using a common structure, which allows for additional test cases to be added to those described herein to accommodate other topologies and/or the modeling of different path characteristics. It is the intention of this work to capture the consensus of the RMCAT working group participants regarding the test cases upon which the performance of the candidate RMCAT proposals should be evaluated.

## 2. Terminology

The terminology defined in RTP [RFC3550], RTP Profile for Audio and Video Conferences with Minimal Control [RFC3551], RTCP Extended Report (XR) [RFC3611], Extended RTP Profile for RTCP-based Feedback (RTP/AVPF) [RFC4585], Support for Reduced-Size RTCP [RFC5506], and RTP Circuit Breaker algorithm [I-D.ietf-avtcore-rtp-circuit-breakers] apply.

## 3. Structure of Test cases

All test cases in this document follow a basic structure allowing implementers to describe a new test scenario without repeatedly explaining common attributes. The structure includes a general description section that describes the test case and its motivation. Additionally the test case defines a set of attributes that characterize the testbed, i.e., the network path between communicating peers and the diverse traffic sources.

### o Define the test case:

- \* General description: describes the motivation and the goals of the test case.
- \* Expected behavior: describe the desired rate adaptation behaviour.
- \* Define a check-list to evaluate the desired behaviour: this indicates the minimum set of metrics (e.g., link utilization, media sending rate) that a proposed algorithm needs to measure to validate the expected rate adaptation behaviour. It should also indicate the time granularity (e.g., averaged over 10ms, 100ms, or 1s) for measuring certain metrics. Typical measurement interval is 200ms.

- o Define testbed topology: every test case needs to define an evaluation testbed topology. Figure 1 shows such an evaluation topology. In this evaluation topology, S1..Sn are traffic sources. These sources generate media traffic and use either an RMCAT candidate congestion control algorithm or other congestion control algorithm designed for media, such as TFRC. R1..Rn are the corresponding receivers. A test case can have one or more such traffic sources (S) and corresponding receivers (R). The path from the source to destination is denoted as forward and the path from a destination to a source is denoted as backward. The following basic structure of test case has been described from the perspective of media generating endpoints attached on the left-hand side of Figure 1. In this setup, media flows in forward direction and corresponding feedback/control messages flow in the backward direction. However, it is also possible to set up the test with media flowing in both forward and backward directions. In that case, unless otherwise specified by the test case, it is expected that the backward path does not introduce any congestion related impairments and has enough capacity to accommodate both media and feedback/control messages. It should be noted that depending on the test cases it is possible to have different path characteristics in of the either directions.

o

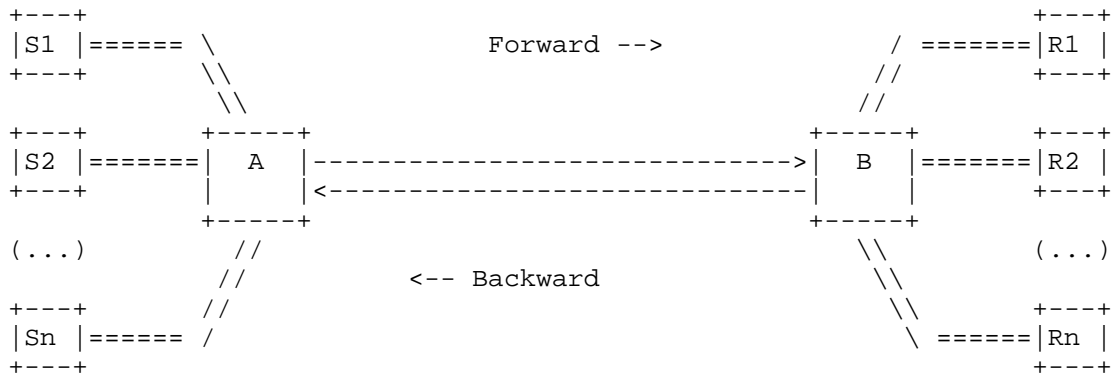


Figure 1: Example of A Testbed Topology

In a laboratory testbed environment there may exist a significant amount of traffic on portions of the network path between the endpoints that is not desired for the purposes of these RMCAT tests. Some of this traffic may be generated by other processes on the endpoints themselves (e.g., discovery protocols) or by other endpoints not presently under test. It is recommended not to route traffic generated by endpoints that are not under test

through the test bed. Additionally, it is recommended to route non-RMCAT traffic generated by the endpoints under test around the bottleneck links specified herein.

o Define testbed attributes:

- \* Duration: defines the duration of the test.
- \* Path characteristics: defines the end-to-end transport level path characteristics of the testbed in a particular test case. Two sets of attributes describe the path characteristics, one for the forward path and the other for the backward path. The path characteristics for a particular path direction is applicable to all the Sources "S" sending traffic on that path. If only one attribute is specified, it is used for both path directions, however, unless specified the reverse path has no capacity restrictions and no path loss.
  - + Path direction: forward or backward.
  - + Bottleneck-link capacity: defines minimum capacity of the end-to-end path
  - + One-way propagation delay: describes the end-to-end latency along the path when network queues are empty, i.e., the time it takes for a packet to go from the sender to the receiver without encountering any queuing delay.
  - + Maximum end-to-end jitter: defines the maximum jitter that can be observed along the path.
  - + Bottleneck queue type: for example, Droptail, FQ-CoDel, or PIE.
  - + Bottleneck queue size: defines size of queue in terms of queuing time when the queue is full (in milliseconds).
  - + Path loss ratio: characterizes the non-congested, additive, losses to be generated on the end-to-end path. MUST describe the loss pattern or loss model used to generate the losses.
- \* Application-related: defines the traffic source behaviour for implementing the test case
  - + Media traffic Source: defines the characteristics of the media sources. When using more than one media source, the

different attributes are enumerated separately for each different media source.

- Media type: Video/Voice/Application/Text
- Media flow direction: forward, backward or both.
- Number of media sources: defines the total number of media sources
- Media codec: Constant Bit Rate (CBR) or Variable Bit Rate (VBR)
- Media source behaviour: describes the media encoder behavior. It defines the main parameters that affect the adaptation behaviour. This may include but not limited to:
  - o Adaptability: describes the adaptation options. For example, in the case of video it defines the following ranges of adaptation: bit rate, frame rate, video resolution. Similarly, in the case of voice, it defines the range of bit rate adaptation, the sampling rate variation, and the variation in packetization interval.
  - o Output variation : for a VBR encoder it defines the encoder output variation from the average target rate over a particular measurement interval. For example, on average the encoder output may vary between 5% to 15% above or below the average target bit rate when measured over a 100 ms time window. The time interval over which the variation is specified must be provided.
  - o Responsiveness to a new bit rate request: the lag in time between a new bit rate request and actual rate changes in encoder output. Depending on the encoder, this value may be specified in absolute time (e.g. 10ms to 1000ms) or other appropriate metric (next frame interval time).
- Media content: describes the chosen media sequences; For example, test sequences are available at: [xiph-seq] and [HEVC-seq].
- Media timeline: describes the point when the media source is introduced and removed from the testbed. For example,

the media source may start transmitting immediately when the test case begins, or after a few seconds.

- Startup behaviour: the media starts at a defined bit rate, which may be the minimum, maximum bit rate, or a value in between (in Kbps).
- + Competing traffic source: describes the characteristics of the competing traffic source, the different types of competing flows are enumerated in [I-D.ietf-rmcat-eval-criteria].
  - Traffic direction: forward, backward or both.
  - Type of sources: defines the types of competing traffic sources. Types of competing traffic flows are listed in [I-D.ietf-rmcat-eval-criteria]. For example, the number of TCP flows connected to a web browser, the mean size and distribution of the content downloaded.
  - Number of sources: defines the total number of competing sources of each media type.
  - Congestion control: enumerates the congestion control used by each type of competing traffic.
  - Traffic timeline: describes when the competing traffic starts and ends in the test case.
- \* Additional attributes: describes attributes essential for implementing a test case which are not included in the above structure. These attributes MUST be well defined, so that other implementers are able to implement it.

Any attribute can have a set of values (enclosed within "[ ]"). Each member value of such a set MUST be treated as different value for the same attribute. It is desired to run separate tests for each such attribute value.

The test cases described in this document follow the above structure.

#### 4. Recommended Evaluation Settings

This section describes recommended test case settings and could be overwritten by the respective test cases.

#### 4.1. Evaluation metrics

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the following metrics at a fine enough time granularity:

1. Flow level:

- A. End-to-end delay for the RMCAT flow.
- B. Variation in sending bit rate and goodput. Mainly observing the frequency and magnitude of oscillations.
- C. Packet losses observed at the receiving endpoint
- D. Feedback message overhead
- E. Convergence time.

2. Transport level:

- A. Bandwidth utilization
- B. Queue length (milliseconds at specified path capacity):
  - + average over the length of the session
  - + 5 and 95 percentile
  - + median, maximum, minimum

#### 4.2. Path characteristics

Each path between a sender and receiver as described in Figure 1 have the following characteristics unless otherwise specified in the test case.

- o Path direction: forward and backward.
- o Bottleneck-link capacity: 4Mbps.
- o One-Way propagation delay: 50ms. It is encouraged to test with additional propagation delays mentioned in Appendix A.1
- o Maximum end-to-end jitter: 30ms. Jitter models are described in Appendix B.1

- o Bottleneck queue type: Drop tail. It is encouraged to test with other AQM schemes, such as FQ-CoDel and PIE.
- o Bottleneck queue size: 300ms.
- o Path loss ratio: 0%.

Examples of additional network parameters are discussed in Appendix A.

#### 4.3. Media source

Unless otherwise specified, each test case will include one or more media sources as described below.

- o Media type: Video
  - \* Media codec: VBR
  - \* Media source behaviour:
    - + Adaptability:
      - Bit rate range: 150 Kbps - 1.5 Mbps. In real-life applications the bitrate range can vary a lot depending on the provided service, for example, the maximum bitrate can be up to 4Mbps. However, for running tests to evaluate the congestion control algorithms it is more important to have a look at how they are reacting to certain amount of bandwidth change. Also it is possible that the media traffic generator used in a particular simulator or testbed is not capable of generating higher bitrate. Hence we have selected a suitable bitrate range typical of consumer-grade video conferencing applications in designing the test case. If a different bitrate range is used in the test cases, the end-to-end path capacity values will also need to be scaled accordingly.
      - Frame resolution: 144p - 720p (or 1080p)
      - Frame rate: 10fps - 30fps
    - + Variation from target bitrate: +/-5%. Unless otherwise specified in the test case, bitrate variation SHOULD be calculated over one (1) second period of time.
    - + Responsiveness to new bit rate request: 100ms

- \* Media content: The media content should represent a typical video conversational scenario with head and shoulder movement. We recommend to use Foreman video sequence.
  - \* Media startup behaviour: 150Kbps. It should be noted that applications can use smart ways to select an optimal startup bitrate values for a certain network condition. In such cases the candidate proposals MAY show the effectiveness of such smart approach as an additional information for the evaluation process.
- o Media type: Audio
    - \* Media codec: CBR
    - \* Media bitrate: 20Kbps

## 5. Basic Test Cases

### 5.1. Variable Available Capacity with Single RMCAT flow

In this test case the bottleneck-link capacity between the two endpoints varies over time. This test is designed to measure the responsiveness of the candidate algorithm. This test tries to address the requirements in [I-D.ietf-rmcat-cc-requirements], which requires the algorithm to adapt the flow(s) and provide lower end-to-end latency when there exists:

- o an intermediate bottleneck
- o change in available capacity (e.g., due to interface change, routing change).
- o maximum Media Bit Rate is Greater than Link Capacity. In this case, the application will attempt to ramp up to its maximum bit rate, since the link capacity is limited to a value lower, the congestion control scheme is expected to stabilize the sending bit rate close to the available bottleneck capacity. This situation can occur when the endpoints are connected via thin long networks even though the advertised capacity of the access network may be higher.

It should be noted that the exact variation in available capacity due to any of the above depends on the under-lying technologies. Hence, we describe a set of known factors, which may be extended to devise a more specific test case targeting certain behaviour in a certain network environment.

Expected behavior: the candidate algorithm is expected to detect the path capacity constraint, converges to bottleneck link's capacity and adapt the flow to avoid unwanted oscillation when the sending bit rate is approaching the bottleneck link's capacity. The oscillations occur when the media flow(s) attempts to reach its maximum bit rate, overshoots the usage of the available bottleneck capacity, to rectify it reduces the bit rate and starts to ramp up again.

Testbed topology: One media source S1 is connected to corresponding R1. The media traffic is transported over the forward path and corresponding feedback/control traffic is transported over the backward path.

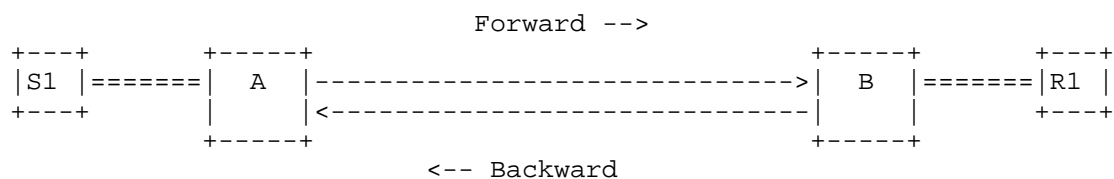


Figure 2: Testbed Topology for Limited Link Capacity

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the metrics described in Section 4.1 at a fine enough time granularity.

Testbed attributes:

- o Test duration: 100s
- o Path characteristics: as described in Section 4.2
- o Application-related:
  - \* Media Traffic:
    - + Media type: Video
      - Media direction: forward.
      - Number of media sources: One (1)
      - Media timeline:
        - o Start time: 0s.
        - o End time: 99s.

- + Media type: Audio
  - Media direction: forward.
  - Number of media sources: One (1)
  - Media timeline:
    - o Start time: 0s.
    - o End time: 99s.
- \* Competing traffic:
  - + Number of sources : Zero (0)
- o Test Specific Information:
  - \* This test uses the following one way propagation delays of 50 ms and 100 ms.
  - \* This test uses bottleneck path capacity variation as listed in Table 1

Variation pattern index	Path direction	Start time	Path capacity
One	Forward	0s	1Mbps
Two	Forward	40s	2.5Mbps
Three	Forward	60s	600Kbps
Four	Forward	80s	1Mbps

Table 1: Path capacity variation pattern for forward direction

## 5.2. Variable Available Capacity with Multiple RMCAT flows

This test case is similar to Section 5.1. However in addition this test will also consider persistent network load due to competing traffic.

Expected behavior: the candidate algorithms is expected to detect the variation in available capacity and adapt the media stream(s) accordingly. The flows stabilize around their maximum bitrate as the as the maximum link capacity is large enough to accommodate the flows. When the available capacity drops, the flow(s) adapts by

decreasing its sending bit rate, and when congestion disappears, the flow(s) are again expected to ramp up.

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the metrics described in Section 4.1 at a fine enough time granularity:

**Testbed Topology:** Two (2) media sources S1 and S2 are connected to their corresponding destinations R1 and R2. The media traffic is transported over the forward path and corresponding feedback/control traffic is transported over the backward path.

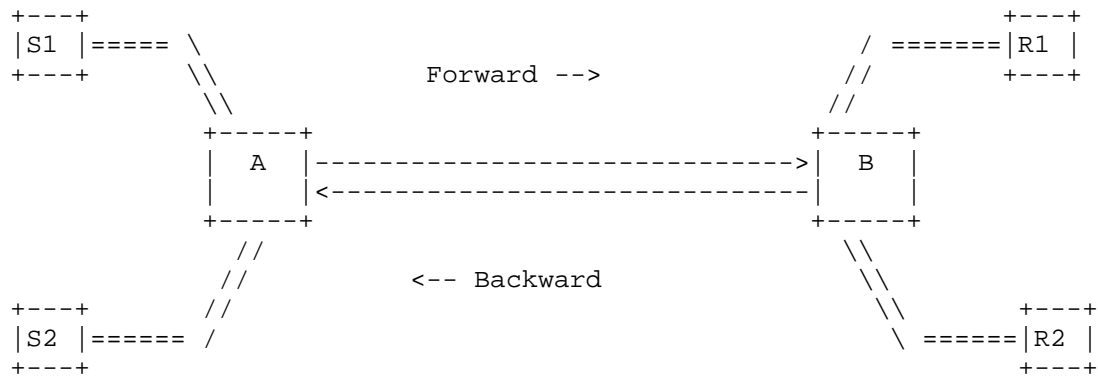


Figure 3: Testbed Topology for Variable Available Capacity

**Testbed attributes:**

Testbed attributes are similar as described in Section 5.1 except the test specific capacity variation setup.

**Test Specific Information:** This test uses path capacity variation as listed in Table 2 with a corresponding end time of 125 seconds.

Variation pattern index	Path direction	Start time	Path capacity
One	Forward	0s	4Mbps
Two	Forward	25s	2Mbps
Three	Forward	50s	3.5Mbps
Four	Forward	75s	1Mbps
Five	Forward	100s	2Mbps

Table 2: Path capacity variation pattern for forward direction

### 5.3. Congested Feedback Link with Bi-directional RMCAT flows

RMCAT WG has been chartered to define algorithms for RTP hence it is assumed that RTCP, RTP header extension or such would be used by the congestion control algorithm in the backchannel. Due to asymmetric nature of the link between communicating peers it is possible for a participating peer to not receive such feedback information due to an impaired or congested backchannel (even when the forward channel might not be impaired). This test case is designed to observe the candidate congestion control behaviour in such an event.

It is expected that the candidate algorithms is able to cope with the lack of feedback information and adapt to minimize the performance degradation of media flows in the forward channel.

It should be noted that for this test case: logs are compared with the reference case, i.e, when the backward channel has no impairments

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the metrics described in Section 4.1 at a fine-grained time intervals:

Testbed topology: One (1) media source S1 is connected to corresponding R1, but both endpoints are additionally receiving and sending data, respectively. The media traffic (S1->R1) is transported over the forward path and corresponding feedback/control traffic is transported over the backward path. Likewise media traffic (S2->R2) is transported over the backward path and corresponding feedback/control traffic is transported over the forward path.

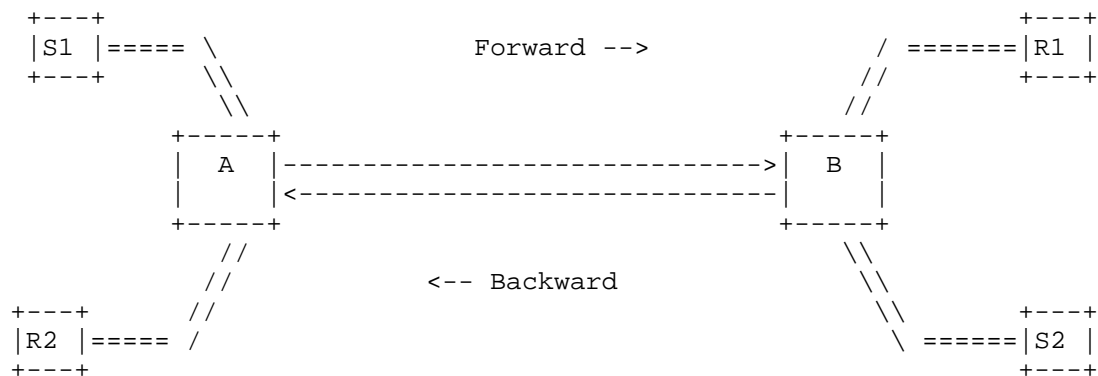


Figure 4: Testbed Topology for Congested Feedback Link

Testbed attributes:

- o Test duration: 100s
- o Path characteristics:
  - \* Bottleneck-link capacity: 2Mbps.
- o Application-related:
  - \* Media Source:
    - + Media type: Video
      - Media direction: forward and backward
      - Number of media sources: Two (2)
      - Media timeline:
        - o Start time: 0s.
        - o End time: 99s.
    - + Media type: Audio
      - Media direction: forward and backward
      - Number of media sources: Two (2)
      - Media timeline:
        - o Start time: 0s.
        - o End time: 99s.
  - \* Competing traffic:
    - + Number of sources : Zero (0)
- o Test Specific Information: This test uses path capacity variations to create congested feedback link. Table 3 lists the variation patterns applied to the forward path and Table 4 lists the variation patterns applied to the backward path.

Variation pattern index	Path direction	Start time	Path capacity
One	Forward	0s	2Mbps
Two	Forward	20s	1Mbps
Three	Forward	40s	500Kbps
Four	Forward	60s	2Mbps

Table 3: Path capacity variation pattern for forward direction

Variation pattern index	Path direction	Start time	Path Capacity
One	Backward	0s	2Mbps
Two	Backward	35s	800Kbps
Three	Backward	70s	2Mbps

Table 4: Path capacity variation pattern for backward direction

#### 5.4. Competing Flows with Same RMCAT Algorithm

In this test case, more than one RMCAT media flow shares the bottleneck link and each of them uses the same congestion control algorithm. This is a typical scenario where a real-time interactive application sends more than one media flows to the same destination and these flows are multiplexed over the same port. In such a scenario it is likely that the flows will be routed via the same path and need to share the available bandwidth amongst themselves. For the sake of simplicity it is assumed that there are no other non-RMCAT competing traffic sources in the bottleneck link and that there is sufficient capacity to accommodate all the flows individually. While this appears to be a variant of the test case defined in Section 5.2, it focuses on the capacity sharing aspect of the candidate algorithm. The previous test case, on the other hand, measures adaptability, stability, and responsiveness of the candidate algorithm.

Expected behavior: It is expected that the competing flows will converge to an optimum bit rate to accommodate all the flows with minimum possible latency and loss. Specifically, the test introduces three media flows at different time instances, when the second flow appears there should still be room to accommodate another flow on the bottleneck link. Lastly, when the third flow appears the bottleneck link should be saturated.

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the metrics described in Section 4.1 at a fine enough time granularity:

Testbed topology: Three media sources S1, S2, S3 are connected to respective R1, R2, R3. The media traffic is transported over the forward path and corresponding feedback/control traffic is transported over the backward path.

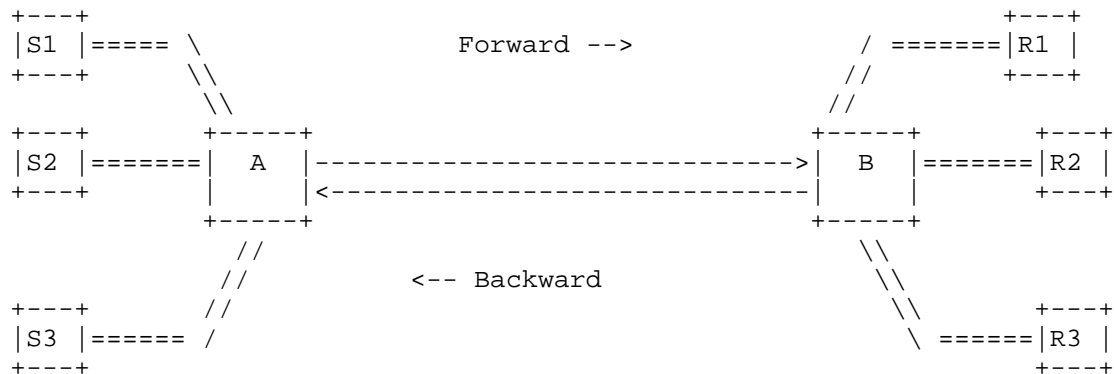


Figure 5: Testbed Topology for Multiple RMCAT Flows

Testbed attributes:

- o Test duration: 120s
- o Path characteristics:
  - \* Bottleneck-link capacity: 3.5Mbps
- o Application-related:
  - \* Media Source:
    - + Media type: Video
      - Media direction: forward.
      - Number of media sources: Three (3)
      - Media timeline: New media flows are added sequentially, at short time intervals. See test specific setup below.
    - + Media type: Audio

- Media direction: forward.
  - Number of media sources: Three (3)
  - Media timeline: New media flows are added sequentially, at short time intervals. See test specific setup below.
- \* Competing traffic:
    - + Number of sources : Zero (0)
- o Test Specific Information:
    - \* Media flow timeline:
      - + Flow ID: One (1)
      - + Start time: 0s
      - + End time: 119s
    - \* Media flow timeline:
      - + Flow ID: Two (2)
      - + Start time: 20s
      - + End time: 119s
    - \* Media flow timeline:
      - + Flow ID: Three (3)
      - + Start time: 40s
      - + End time: 119s

#### 5.5. Round Trip Time Fairness

In this test case, multiple RMCAT media flows share the bottleneck link, but the end-to-end path latency for each RMCAT flow is different. For the sake of simplicity it is assumed that there are no other non-RMCAT competing traffic sources in the bottleneck link and that there is sufficient capacity to accommodate all the flows. While this appears to be a variant of test case 5.2, it focuses on the capacity sharing aspect of the candidate algorithm under different RTTs.

It is expected that the competing flows will converge to bit rates to accommodate all the flows with minimum possible latency and loss. Specifically, the test introduces five media flows at the same time instance.

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the metrics described in Section 4.1 at a fine enough time granularity:

Testbed Topology: Five (5) media sources  $S_1, S_2, \dots, S_5$  are connected to their corresponding media sinks  $R_1, R_2, \dots, R_5$ . The media traffic is transported over the forward path and corresponding feedback/control traffic is transported over the backward path. The topology is the same as in Section 5.4. The end-to-end path delays are: 10ms for  $S_1-R_1$ , 25ms for  $S_2-R_2$ , 50ms for  $S_3-R_3$ , 100ms for  $S_4-R_4$ , and 150ms  $S_5-R_5$ , respectively.

Testbed attributes:

- o Test duration: 120s
- o Path characteristics:
  - \* One-Way propagation delay for each flow: 10ms, 25ms, 50ms, 100ms, 150ms.
- o Application-related:
  - \* Media Source:
    - + Media type: Video
      - Media direction: forward
      - Number of media sources: Five (5)
      - Media timeline:
        - o Start time: 0s.
        - o End time: 119s.
    - + Media type: Audio
      - Media direction: forward.
      - Number of media sources: Five (5)

- Media timeline:
  - o Start time: 0s.
  - o End time: 119s.
- \* Competing traffic:
  - + Number of sources : Zero (0)
- o Test Specific Information: None

#### 5.6. RMCAT Flow competing with a long TCP Flow

In this test case, one or more RMCAT media flows share the bottleneck link with at least one long lived TCP flows. Long lived TCP flows download data throughout the session and are expected to have infinite amount of data to send and receive. This is a scenario where a multimedia application co-exists with a large file download. The test case measures the adaptivity of the candidate algorithm to competing traffic. It addresses the requirement 3 in [I-D.ietf-rmc-cat-requirements].

Expected behavior: depending on the convergence observed in test case 5.1 and 5.2, the candidate algorithm may be able to avoid congestion collapse. In the worst case, the media stream will fall to the minimum media bit rate.

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the following metrics in addition to the metrics described in Section 4.1 at a fine enough time granularity:

##### 1. Flow level:

###### A. TCP throughput.

Testbed topology: One (1) media source S1 is connected to corresponding media sink, R1. In addition, there is a long-live TCP flow sharing the same bottleneck link. The media traffic is transported over the forward path and corresponding feedback/control traffic is transported over the backward path. The TCP traffic goes over the forward path from, S\_tcp with acknowledgement packets flowing along the backward path from, R\_tcp.

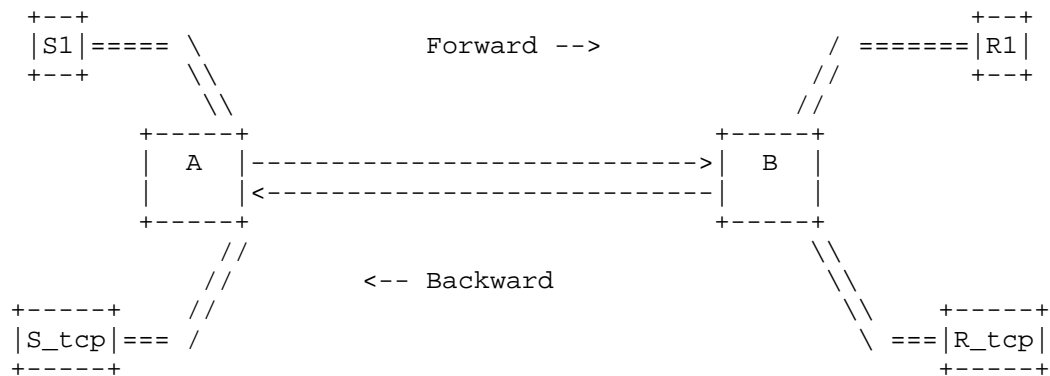


Figure 6: Testbed Topology for TCP vs RMCAT Flows

## Testbed attributes:

- o Test duration: 120s
- o Path characteristics:
  - \* Bottleneck-link capacity: 2Mbps
  - \* Bottleneck queue size: [20ms, 300ms, 1000ms]
- o Application-related:
  - \* Media Source:
    - + Media type: Video
      - Media direction: forward
      - Number of media sources: One (1)
      - Media timeline:
        - o Start time: 5s.
        - o End time: 119s.
    - + Media type: Audio
      - Media direction: forward
      - Number of media sources: One (1)

- Media timeline:
    - o Start time: 5s.
    - o End time: 119s.
  - \* Additionally, implementers are encouraged to run the experiment with multiple media sources.
  - \* Competing traffic:
    - + Number and Types of sources : one (1), long-lived TCP
    - + Traffic direction : forward
    - + Congestion control: Default TCP congestion control.
    - + Traffic timeline:
      - Start time: 0s.
      - End time: 119s.
  - o Test Specific Information: None
- ### 5.7. RMCAT Flow competing with short TCP Flows

In this test case, one or more RMCAT media flow shares the bottleneck link with multiple short-lived TCP flows. Short-lived TCP flows resemble the on/off pattern observed in the web traffic, wherein clients (browsers) connect to a server and download a resource (typically a web page, few images, text files, etc.) using several TCP connections (up to 4). This scenario shows the performance of the multimedia application when several browser windows are active. The test case measures the adaptivity of the candidate algorithm to competing web traffic, it addresses the requirements 1.E in [I-D.ietf-rmcat-cc-requirements].

Depending on the number of short TCP flows, the cross-traffic either appears as a short burst flow or resembles a long TCP flow. The intention of this test is to observe the impact of short-term burst on the behaviour of the candidate algorithm.

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the following metrics in addition to the metrics described in Section 4.1 at a fine enough time granularity:

1. Flow level:

- A. Variation in the sending rate of the TCP flow.
- B. TCP throughput.

Testbed topology: The topology described here is same as the one described in Figure 6.

Testbed attributes:

- o Test duration: 300s
- o Path characteristics:
  - \* Bottleneck-link capacity: 2.0Mbps
- o Application-related:
  - \* Media Source:
    - + Media type: Video
      - Media direction: forward
      - Number of media sources: two (2)
      - Media timeline:
        - o Start time: 5s.
        - o End time: 299s.
    - + Media type: Audio
      - Media direction: forward
      - Number of media sources: two (2)
      - Media timeline:
        - o Start time: 5s.
        - o End time: 299s.
  - \* Competing traffic:

- + Number and Types of sources : Ten (10), short-lived TCP flows.
  - + Traffic direction : forward
  - + Congestion algorithm: Default TCP Congestion control.
  - + Traffic timeline: Each short TCP flow is modeled as a sequence of file downloads interleaved with idle periods. See test specific setup. Not all short TCPs start at the same time, 2 start in the ON state while 8 start in an OFF state. The model for the idle times for the OFF state is discussed in the Short-TCP model.
- o Test Specific Information:
    - \* Short-TCP traffic model:
      - + File sizes: uniform distribution between 100KB to 1MB
      - + Idle period: the duration of the OFF state is derived from an exponential distribution with the mean value of 10 seconds.

#### 5.8. Media Pause and Resume

In this test case, more than one real-time interactive media flows share the link bandwidth and all flows reach to a steady state by utilizing the link capacity in an optimum way. At these stage one of the media flow is paused for a moment. This event will result in more available bandwidth for the rest of the flows and as they are on a shared link. When the paused media flow will resume it would no longer have the same bandwidth share on the link. It has to make it's way through the other existing flows in the link to achieve a fair share of the link capacity. This test case is important specially for real-time interactive media which consists of more than one media flows and can pause/resume media flow at any point of time during the session. This test case directly addresses the requirement number 5 in [I-D.ietf-rmcat-cc-requirements]. One can think it as a variation of test case defined in Section 5.4. However, it is different as the candidate algorithms can use different strategies to increase its efficiency, for example the fairness, convergence time, reduce oscillation etc, by capitalizing the fact that they have previous information of the link.

To evaluate the performance of the candidate algorithms it is expected to log enough information to visualize the following metrics

in addition to the metrics described in Section 4.1 at a fine enough time granularity:

1. Flow level:

- A. Variation in sending bit rate and goodput. Mainly observing the frequency and magnitude of oscillations.

Testbed Topology: Same as test case defined in Section 5.4

Testbed attributes: The general description of the testbed parameters are same as Section 5.4 with changes in the test specific setup as below-

o Other test specific setup:

- \* Media flow timeline:

- + Flow ID: One (1)
- + Start time: 0s
- + Flow duration: 119s
- + Pause time: not required
- + Resume time: not required

- \* Media flow timeline:

- + Flow ID: Two (2)
- + Start time: 0s
- + Flow duration: 119s
- + Pause time: at 40s
- + Resume time: at 60s

- \* Media flow timeline:

- + Flow ID: Three (3)
- + Start time: 0s
- + Flow duration: 119s

- + Pause time: not required
- + Resume time: not required

## 6. Other potential test cases

It has been noticed that there are other interesting test cases besides the basis test cases listed above. In many aspects, these additional test cases can help to further evaluate the candidate algorithm. They are listed as below.

### 6.1. Explicit Congestion Notification Usage

This test case requires to run all the basic test cases with the availability of Explicit Congestion Notification (ECN) [RFC6679] feature enabled. The goal of this test is to exhibit that the candidate algorithms does not fail when ECN signals are available. With ECN signals enabled the algorithms are expected to perform better than their delay based variants.

### 6.2. Multiple Bottlenecks

In this test case one RMCAT flow, S1->R2 traverse a path with multiple bottlenecks. As illustrated in Figure 7, the first flow (S1->R1) competes with the second RMCAT flow (S2->R2) over the link between A and B which is close to the sender side; again, that flow (S1->R1) competes with the third RMCAT flow (S3->R3) over the link between C and D which is close to the receiver side. The goal of this test is to ensure that the candidate algorithms work properly in the presence of multiple bottleneck links on the end to end path.

Expected behavior: the candidate algorithm is expected to achieve full utilization at both bottleneck links without starving any of the three RMCAT flows.

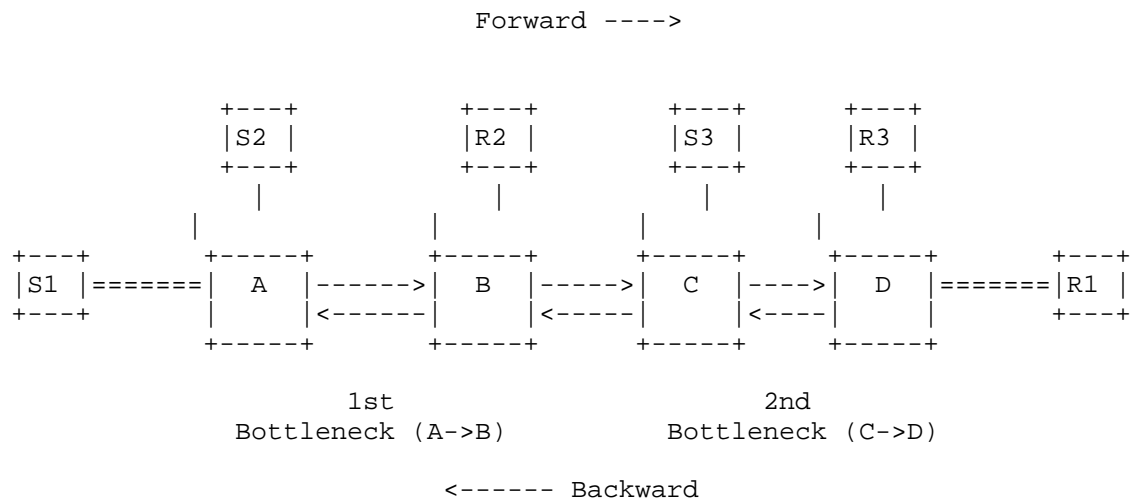


Figure 7: Testbed Topology for Multiple Bottlenecks

Testbed topology: Three media sources S1, S2, and S3 are connected to respective destinations R1, R2, and R3. For all three flows the media traffic is transported over the forward path and corresponding feedback/control traffic is transported over the backward path.

Testbed attributes:

- o Test duration: 120s
- o Path characteristics:
  - \* Path capacity between A and B = 2Mbps.
  - \* Path capacity between B and C = 8Mbps.
  - \* Path capacity between C and D = 1.5Mbps.
  - \* One-Way propagation delay:
    1. Between S1 and R1 : 100ms
    2. Between S2 and R2: 40ms
    3. Between S3 and R3: 40ms

- o Application-related:
  - \* Media Source:
    - + Media type: Video
      - Media direction: Forward
      - Number of media sources: Three (3)
      - Media timeline:
        - o Start time: 0s.
        - o End time: 119s.
    - + Media type: Audio
      - Media direction: Forward
      - Number of media sources: Three (3)
      - Media timeline:
        - o Start time: 0s.
        - o End time: 119s.
  - \* Competing traffic:
    - + Number of sources : Zero (0)

## 7. Wireless Access Links

### 7.1. Cellular Network Specific Test Cases

Additional cellular network specific test cases are define in  
[I-D.draft-sarker-rmcat-cellular-eval-test-cases]

### 7.2. Wi-Fi Network Specific Test Cases

TBD

[Editor's Note: We should encourage people to come up with possible  
WiFi Network specific test cases]

## 8. Security Considerations

Security issues have not been discussed in this memo.

## 9. IANA Considerations

There are no IANA impacts in this memo.

## 10. Acknowledgements

Much of this document is derived from previous work on congestion control at the IETF.

The content and concepts within this document are a product of the discussion carried out in the Design Team.

## 11. References

### 11.1. Normative References

- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

- [I-D.ietf-avtcore-rtp-circuit-breakers]  
Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", draft-ietf-avtcore-rtp-circuit-breakers-05 (work in progress), February 2014.
- [I-D.ietf-rmcat-eval-criteria]  
Singh, V. and J. Ott, "Evaluating Congestion Control for Interactive Real-time Media", draft-ietf-rmcat-eval-criteria-00 (work in progress), January 2014.
- [I-D.ietf-rmcat-cc-requirements]  
Jesup, R., "Congestion Control Requirements For RMCAT", draft-ietf-rmcat-cc-requirements-02 (work in progress), February 2014.
- [I-D.draft-sarker-rmcat-cellular-eval-test-cases]  
Sarker, Z., "Evaluation Test Cases for Interactive Real-Time Media over Cellular Networks", <<http://www.ietf.org/id/draft-sarker-rmcat-cellular-eval-test-cases-00.txt>>.

## 11.2. Informative References

- [I-D.ietf-rtcweb-use-cases-and-requirements]  
Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-14 (work in progress), February 2014.
- [RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, August 2007.
- [RFC5166] Floyd, S., "Metrics for the Evaluation of Congestion Control Mechanisms", RFC 5166, March 2008.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [SA4-EVAL]  
R1-081955, 3GPP., "LTE Link Level Throughput Data for SA4 Evaluation Framework", 3GPP R1-081955, 5 2008.
- [SA4-LR]  
S4-050560, 3GPP., "Error Patterns for MBMS Streaming over UTRAN and GERAN", 3GPP S4-050560, 5 2008.
- [xiph-seq]  
Xiph.org, , "Video Test Media", <http://media.xiph.org/video/derf/> , .

## [HEVC-seq]

HEVC, , "Test Sequences",  
[http://www.netlab.tkk.fi/~varun/test\\_sequences/](http://www.netlab.tkk.fi/~varun/test_sequences/) , .

## [TCP-eval-suite]

Lachlan, A., Marcondes, C., Floyd, S., Dunn, L., Guillier, R., Gang, W., Eggert, L., Ha, S., and I. Rhee, "Towards a Common TCP Evaluation Suite", Proc. PFLDnet. 2008, August 2008.

## Appendix A. List of Network Parameters

In addition to the recommended evaluation settings in Section 4, the implementors can extend their tests by choosing from the following values:

## A.1. One-way Propagation Delay

Experiments are expected to verify that the congestion control is able to work in challenging situations, for example over trans-continental and/or satellite links. Typical values are:

1. Very low latency: 0-1ms
2. Low latency: 50ms
3. High latency: 150ms
4. Extreme latency: 300ms

## A.2. End-to-end Loss

To model lossy links, the experiments can choose one of the following loss rates, the fractional loss is the ratio of packets lost and packets sent.

1. no loss: 0%
2. 1%
3. 5%
4. 10%
5. 20%

### A.3. DropTail Router Queue Length

The router queue length is measured as the time taken to drain the FIFO queue. It has been noted in various discussions that the queue length in the current deployed Internet varies significantly. While the core backbone network has very short queue length, the home gateways usually have larger queue length. Those various queue lengths can be categorized in the following way:

1. QoS-aware (or short): 70ms
2. Nominal: 300-500ms
3. Buffer-bloated: 1000-2000ms

Here the size of the queue is measured in bytes or packets and to convert the queue length measured in seconds to queue length in bytes:

$$\text{QueueSize (in bytes)} = \text{QueueSize (in sec)} \times \text{Throughput (in bps)} / 8$$

## Appendix B. Models

### B.1. Jitter models

This section defines jitter model for the purposes of this document. When jitter is to be applied to both the RMCAT flow and any competing flow (such as a TCP competing flow), the competing flow will use the jitter definition below that does not allow for re-ordering of packets on the competing flow (see NR-RBPDV definition below).

Jitter is an overloaded term in communications. Its meaning is typically associated with the variation of a metric (e.g., delay) with respect to some reference metric (e.g., average delay or minimum delay). For example, RFC 3550 jitter is a smoothed estimate of jitter which is particularly meaningful if the underlying packet delay variation was caused by a Gaussian random process.

Because jitter is an overloaded term, we instead use the term Packet Delay Variation (PDV) to describe the variation of delay of individual packets in the same sense as the IETF IPPM WG has defined PDV in their documents (e.g., RFC 3393) and as the ITU-T SG16 has defined IP Packet Delay Variation (IPDV) in their documents (e.g., Y.1540).

Most PDV distributions in packet network systems are one-sided distributions (the measurement of which with a finite number of measurement samples result in one-sided histograms). In the usual

packet network transport case there is typically one packet that transited the network with the minimum delay, then a majority of packets also transit the system within some variation from this minimum delay, and then a minority of the packets transits the network with delays higher than the median or average transit time (these are outliers). Although infrequent, outliers can cause significant deleterious operation in adaptive systems and should be considered in RMCAT adaptation designs.

In this section we define two different bounded PDV characteristics, 1) Random Bounded PDV and 2) Approximately Random Subject to No-Reordering Bounded PDV.

Random Bounded PDV (RBPDV):

The RBPDV probability distribution function (pdf) is specified to be of some mathematically describable function which includes some practical minimum and maximum discrete values suitable for testing. For example, the minimum value,  $x_{\min}$ , might be specified as the minimum transit time packet and the maximum value,  $x_{\max}$ , might be defined to be two standard deviations higher than the mean.

Since we are typically interested in the distribution relative to the mean delay packet, we define the zero mean PVD sample,  $z(n)$ , to be  $z(n) = x(n) - x_{\text{mean}}$ , where  $x(n)$  is a sample of the RBPDV random variable  $x$  and  $x_{\text{mean}}$  is the mean of  $x$ .

We assume here that  $s(n)$  is the original source time of packet  $n$  and the post-jitter induced emission time,  $j(n)$ , for packet  $n$  is  $j(n) = \{[z(n) + x_{\text{mean}}] + s(n)\}$ . It follows that the separation in the post-jitter time of packets  $n$  and  $n+1$  is  $\{[s(n+1)-s(n)] - [z(n)-z(n+1)]\}$ . Since the first term is always a positive quantity, we note that packet reordering at the receiver is possible whenever the second term is greater than the first. Said another way, whenever the difference in possible zero mean PDV sample delays (i.e.,  $[x_{\max}-x_{\min}]$ ) exceeds the inter-departure time of any two sent packets, we have the possibility of packet re-ordering.

There are important use cases in real networks where packets can become re-ordered such as in load balancing topologies and during route changes. However, for the vast majority of cases there is no packet re-ordering because most of the time packets follow the same path. Due to this, if a packet becomes overly delayed, the packets after it on that flow are also delayed. This is especially true for mobile wireless links where there are per-flow queues prior to base station scheduling. Owing to this important use case, we define another PDV profile similar to the above, but one that does not allow for re-ordering within a flow.

Approximately Random Subject to No-Reordering Bounded PDV (NR-RPVD):

No Reordering RPDV, NR-RPVD, is defined similarly to the above with one important exception. Let  $\text{serial}(n)$  be defined as the serialization delay of packet  $n$  at the lowest bottleneck link rate (or other appropriate rate) in a given test. Then we produce all the post-jitter values for  $j(n)$  for  $n = 1, 2, \dots, N$ , where  $N$  is the length of the source sequence  $s$  to be jittered. The exception can be stated as follows: We revisit all  $j(n)$  beginning from index  $n=2$ , and if  $j(n)$  is determined to be less than  $[j(n-1)+\text{serial}(n-1)]$ , we redefine  $j(n)$  to be equal to  $[j(n-1)+\text{serial}(n-1)]$  and continue for all remaining  $n$  (i.e.,  $n = 3, 4, \dots, N$ ). This models the case where the packet  $n$  is sent immediately after packet  $(n-1)$  at the bottleneck link rate. Although this is generally the theoretical minimum in that it assumes that no other packets from other flows are in-between packet  $n$  and  $n+1$  at the bottleneck link, it is a reasonable assumption for per flow queuing.

We note that this assumption holds for some important exception cases, such as packets immediately following outliers. There are a multitude of software controlled elements common on end-to-end Internet paths (such as firewalls, ALGs and other middleboxes) which stop processing packets while servicing other functions (e.g., garbage collection). Often these devices do not drop packets, but rather queue them for later processing and cause many of the outliers. Thus NR-RPVD models this particular use case (assuming  $\text{serial}(n+1)$  is defined appropriately for the device causing the outlier) and thus is believed to be important for adaptation development for RMCAT.

[Editor's Note: It may require to define test distributions as well. Example test distribution may include-

1 - Two-sided: Uniform PDV Distribution. Two quantities to define:  $x_{\min}$  and  $x_{\max}$ .

2 - Two-sided: Truncated Gaussian PDV Distribution. Four quantities to define: the appropriate  $x_{\min}$  and  $x_{\max}$  for test (e.g.,  $\pm$  two sigma values), the standard deviation and the mean.

3 - One Sided: TBD]

## B.2. Loss generation model

[Editor's note : Describes the model for generating packet losses, for example, losses can be generated using traces, or using the Gilbert-Elliott model, or randomly (uncorrelated loss).]

### B.3. TCP traffic model

Long-lived TCP flows will download data throughout the session and are expected to have infinite amount of data to send or receive.

Each short TCP flow is modeled as a sequence of file downloads interleaved with idle periods. Not all short TCPs start at the same time, i.e., some start in the ON state while others start in the OFF state.

The short TCP flows can be modelled in two ways, 1) 100s of flows fetching small (5-20 KB) amounts of data, or 2) 10s of flows fetching slightly larger (100-1000KB) amounts of data.

The idle period is typically derived from an exponential distribution with the mean value of 10 seconds.

[Open issue: short-lived/bursty TCP cross-traffic parameters are still to be agreed upon].

#### Authors' Addresses

Zaheduzzaman Sarker  
Ericsson AB  
Lulea, SE 977 53  
Sweden

Phone: +46 10 717 37 43  
Email: zaheduzzaman.sarker@ericsson.com

Varun Singh  
Aalto University  
School of Electrical Engineering  
Otakaari 5 A  
Espoo, FIN 02150  
Finland

Email: varun@comnet.tkk.fi  
URI: <http://www.netlab.tkk.fi/~varun/>

Xiaoqing Zhu  
Cisco Systems  
510 McCarthy Blvd  
Milpitas, CA 95134  
USA

Email: xiaoqzhu@cisco.com

Michael A. Ramalho  
Cisco Systems, Inc.  
8000 Hawkins Road  
Sarasota, FL 34241  
USA

Phone: +1 919 476 2038  
Email: mramalho@cisco.com

RTP Media Congestion Avoidance  
Techniques (rmcat)  
Internet-Draft  
Intended status: Experimental  
Expires: November 8, 2014

M. Welzl  
S. Islam  
S. Gjessing  
University of Oslo  
May 7, 2014

Coupled congestion control for RTP media  
draft-welzl-rmcat-coupled-cc-03

Abstract

When multiple congestion controlled RTP sessions traverse the same network bottleneck, it can be beneficial to combine their controls such that the total on-the-wire behavior is improved. This document describes such a method for flows that have the same sender, in a way that is as flexible and simple as possible while minimizing the amount of changes needed to existing RTP applications.

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 November 8, 2014.

Copyright Notice

Copyright (c) 2014 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. Definitions . . . . .	3
3. Limitations . . . . .	4
4. Architectural overview . . . . .	5
5. Roles . . . . .	6
5.1. SBD . . . . .	6
5.2. FSE . . . . .	6
5.3. Flows . . . . .	7
5.3.1. Example algorithm 1 - Active FSE . . . . .	7
5.3.2. Example algorithm 2 - Conservative Active FSE . . . . .	8
6. Acknowledgements . . . . .	9
7. IANA Considerations . . . . .	10
8. Security Considerations . . . . .	10
9. References . . . . .	10
9.1. Normative References . . . . .	10
9.2. Informative References . . . . .	11
Appendix A. Example algorithm - Passive FSE . . . . .	11
A.1. Example operation (passive) . . . . .	14
Appendix B. Change log . . . . .	18
B.1. Changes from -00 to -01 . . . . .	18
B.2. Changes from -01 to -02 . . . . .	18
B.3. Changes from -02 to -03 . . . . .	18
Authors' Addresses . . . . .	18

## 1. Introduction

When there is enough data to send, a congestion controller must increase its sending rate until the path's capacity has been reached; depending on the controller, sometimes the rate is increased further, until packets are ECN-marked or dropped. This process inevitably creates undesirable queuing delay -- an effect that is amplified when multiple congestion controlled connections traverse the same network bottleneck. When such connections originate from the same host, it would therefore be ideal to use only one single sender-side congestion controller which determines the overall allowed sending rate, and then use a local scheduler to assign a proportion of this rate to each RTP session. This way, priorities could also be implemented quite easily, as a function of the scheduler; honoring user-specified priorities is, for example, required by rtcweb [rtcweb-usecases].

The Congestion Manager (CM) [RFC3124] provides a single congestion controller with a scheduling function just as described above. It is hard to implement because it requires an additional congestion controller and removes all per-connection congestion control functionality, which is quite a significant change to existing RTP based applications. This document presents a method that is easier to implement than the CM and also requires less significant changes to existing RTP based applications. It attempts to roughly approximate the CM behavior by sharing information between existing congestion controllers, akin to "Ensemble Sharing" in [RFC2140].

## 2. Definitions

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].

### Available Bandwidth:

The available bandwidth is the nominal link capacity minus the amount of traffic that traversed the link during a certain time interval, divided by that time interval.

### Bottleneck:

The first link with the smallest available bandwidth along the path between a sender and receiver.

### Flow:

A flow is the entity that congestion control is operating on. It could, for example, be a transport layer connection, an RTP session, or a subsession that is multiplexed onto a single RTP

session together with other subsessions.

Flow Group Identifier (FGI):

A unique identifier for each subset of flows that is limited by a common bottleneck.

Flow State Exchange (FSE):

The entity that maintains information that is exchanged between flows.

Flow Group (FG):

A group of flows having the same FGI.

Shared Bottleneck Detection (SBD):

The entity that determines which flows traverse the same bottleneck in the network, or the process of doing so.

### 3. Limitations

Sender-side only:

Coupled congestion control as described here only operates inside a single host on the sender side. This is because, irrespective of where the major decisions for congestion control are taken, the sender of a flow needs to eventually decide the transmission rate. Additionally, the necessary information about how much data an application can currently send on a flow is typically only available at the sender side, making the sender an obvious choice for placement of the elements and mechanisms described here.

When implementing a sender-side change to a congestion control mechanism such as TFRC [RFC5348], where receiver-side calculations make assumptions about the rate of the sender, the receiver also needs to be updated accordingly. Flows that have different senders but the same receiver, or different senders and different receivers can also share a bottleneck; such scenarios have been omitted for simplicity, and could be incorporated in future versions of this document. Note that limiting the number of flows on which coupled congestion control operates merely limits the benefits derived from the mechanism.

Shared bottlenecks do not change quickly:

As per the definition above, a bottleneck depends on cross traffic, and since such traffic can heavily fluctuate, bottlenecks can change at a high frequency (e.g., there can be oscillation between two or more links). This means that, when

flows are partially routed along different paths, they may quickly change between sharing and not sharing a bottleneck. For simplicity, here it is assumed that a shared bottleneck is valid for a time interval that is significantly longer than the interval at which congestion controllers operate. Note that, for the only SBD mechanism defined in this document (multiplexing on the same five-tuple), the notion of a shared bottleneck stays correct even in the presence of fast traffic fluctuations: since all flows that are assumed to share a bottleneck are routed in the same way, if the bottleneck changes, it will still be shared.

#### 4. Architectural overview

Figure 1 shows the elements of the architecture for coupled congestion control: the Flow State Exchange (FSE), Shared Bottleneck Detection (SBD) and Flows. The FSE is a storage element that can be implemented in two ways: active and passive. In the active version, it initiates communication with flows and SBD. However, in the passive version, it does not actively initiate communication with flows and SBD; its only active role is internal state maintenance (e.g., an implementation could use soft state to remove a flow's data after long periods of inactivity). Every time a flow's congestion control mechanism would normally update its sending rate, the flow instead updates information in the FSE and performs a query on the FSE, leading to a sending rate that can be different from what the congestion controller originally determined. Using information about/from the currently active flows, SBD updates the FSE with the correct Flow State Identifiers (FSIs).

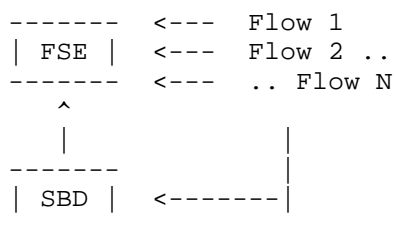


Figure 1: Coupled congestion control architecture

Since everything shown in Figure 1 is assumed to operate on a single host (the sender) only, this document only describes aspects that have an influence on the resulting on-the-wire behavior. It does,

for instance, not define how many bits must be used to represent FSIs, or in which way the entities communicate. Implementations can take various forms: for instance, all the elements in the figure could be implemented within a single application, thereby operating on flows generated by that application only. Another alternative could be to implement both the FSE and SBD together in a separate process which different applications communicate with via some form of Inter-Process Communication (IPC). Such an implementation would extend the scope to flows generated by multiple applications. The FSE and SBD could also be included in the Operating System kernel.

## 5. Roles

This section gives an overview of the roles of the elements of coupled congestion control, and provides an example of how coupled congestion control can operate.

### 5.1. SBD

SBD uses knowledge about the flows to determine which flows belong in the same Flow Group (FG), and assigns FGIs accordingly. This knowledge can be derived from measurements, by considering correlations among measured delay and loss as an indication of a shared bottleneck, or it can be based on the simple assumption that packets sharing the same five-tuple (IP source and destination address, protocol, and transport layer port number pair) and having the same Differentiated Services Code Point (DSCP) in the IP header are typically treated in the same way along the path. The latter method is the only one specified in this document: SBD MAY consider all flows that use the same five-tuple and DSCP to belong to the same FG. This classification applies to certain tunnels, or RTP flows that are multiplexed over one transport (cf. [transport-multiplex]). In one way or another, such multiplexing will probably be recommended for use with rtcweb [rtcweb-rtp-usage].

### 5.2. FSE

The FSE contains a list of all flows that have registered with it. For each flow, it stores the following:

- o a unique flow number to identify the flow
- o the FGI of the FG that it belongs to (based on the definitions in this document, a flow has only one bottleneck, and can therefore be in only one FG)

- o a priority P, which here is assumed to be represented as a floating point number in the range from 0.1 (unimportant) to 1 (very important). A negative value is used to indicate that a flow has terminated
- o The rate used by the flow, FSE\_R

In the FSE, each FG contains one static variable S\_CR which is meant to be the sum of the calculated rates of all flows in the same FG (including the flow itself). This value is used to calculate the sending rate.

The information listed here is enough to implement the sample flow algorithm given below. FSE implementations could easily be extended to store, e.g., a flow's current sending rate for statistics gathering or future potential optimizations.

### 5.3. Flows

Flows register themselves with SBD and FSE when they start, deregister from the FSE when they stop, and carry out an UPDATE function call every time their congestion controller calculates a new sending rate. Via UPDATE, they provide the newly calculated rate and the desired rate (less than the calculated rate in case of application-limited flows, the same otherwise).

Below, two example algorithms are described. While other algorithms could be used instead, the same algorithm must be applied to all flows.

#### 5.3.1. Example algorithm 1 - Active FSE

This algorithm was designed to be the simplest possible method to assign rates according to the priorities of flows. Simulations results in [tech-report-fse] indicate that it does however not significantly reduce queuing delay and packet loss.

- (1) When a flow f starts, it registers itself with SBD and the FSE. FSE\_R is initialized with the congestion controller's initial rate. SBD will assign the correct FGI. When a flow is assigned an FGI, it adds its FSE\_R to S\_CR.
- (2) When a flow f stops, its entry is removed from the list.
- (3) Every time the congestion controller of the flow f determines a new sending rate CC\_R, the flow calls UPDATE, which carries out the tasks listed below to derive the new sending rates for all the flows in the FG. A flow's UPDATE function uses a local

(i.e. per-flow) temporary variable  $S_P$ , which is the sum of all the priorities.

(a) It updates  $S_{CR}$ .

$$S_{CR} = S_{CR} + CC_R - FSE_R(f)$$

(b) It calculates the sum of all the priorities,  $S_P$ .

```

S_P = 0
for all flows i in FG do
    S_P = S_P + P(i)
end for

```

(c) It calculates the sending rates for all the flows in an FG and distributes them.

```

for all flows i in FG do
    FSE_R(i) = (P(i)*S_CR)/S_P
    send FSE_R(i) to the flow i
end for

```

### 5.3.2. Example algorithm 2 - Conservative Active FSE

This algorithm extends algorithm 1 to conservatively emulate the behavior of a single flow by proportionally reducing the aggregate rate on congestion. Simulations results in [tech-report-fse] indicate that it can significantly reduce queuing delay and packet loss.

- (1) When a flow  $f$  starts, it registers itself with SBD and the FSE.  $FSE_R$  is initialized with the congestion controller's initial rate. SBD will assign the correct FGI. When a flow is assigned an FGI, it adds its  $FSE_R$  to  $S_{CR}$ .
- (2) When a flow  $f$  stops, its entry is removed from the list.
- (3) Every time the congestion controller of the flow  $f$  determines a new sending rate  $CC_R$ , the flow calls UPDATE, which carries out the tasks listed below to derive the new sending rates for all the flows in the FG. A flow's UPDATE function uses a local (i.e. per-flow) temporary variable  $S_P$ , which is the sum of all the priorities, and a local variable DELTA, which is used to calculate the difference between  $CC_R$  and the previously stored  $FSE_R$ . To prevent flows from either ignoring congestion or

overreacting, a timer keeps them from changing their rates immediately after the common rate reduction that follows a congestion event. This timer is set to 2 RTTs of the flow that experienced congestion because it is assumed that a congestion event can persist for up to one RTT of that flow, with another RTT added to compensate for fluctuations in the measured RTT value.

(a) It updates  $S\_CR$  based on  $DELTA$ .

```
if Timer has expired or not set then
  DELTA = CC_R - FSE_R(f)
  if DELTA < 0 then // Reduce S_CR proportionally
    S_CR = S_CR * CC_R / FSE_R(f)
    Set Timer for 2 RTTs
  else
    S_CR = S_CR + DELTA
  end if
end if
```

(b) It calculates the sum of all the priorities,  $S\_P$ .

```
S_P = 0
for all flows i in FG do
  S_P = S_P + P(i)
end for
```

(c) It calculates the sending rates for all the flows in an FG and distributes them.

```
for all flows i in FG do
  FSE_R(i) = (P(i)*S_CR)/S_P
  send FSE_R(i) to the flow i
end for
```

## 6. Acknowledgements

This document has benefitted from discussions with and feedback from David Hayes, Andreas Petlund, and David Ros (who also gave the FSE its name).

This work was partially funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700).

## 7. IANA Considerations

This memo includes no request to IANA.

## 8. Security Considerations

In scenarios where the architecture described in this document is applied across applications, various cheating possibilities arise: e.g., supporting wrong values for the calculated rate, the desired rate, or the priority of a flow. In the worst case, such cheating could either prevent other flows from sending or make them send at a rate that is unreasonably large. The end result would be unfair behavior at the network bottleneck, akin to what could be achieved with any UDP based application. Hence, since this is no worse than UDP in general, there seems to be no significant harm in using this in the absence of UDP rate limiters.

In the case of a single-user system, it should also be in the interest of any application programmer to give the user the best possible experience by using reasonable flow priorities or even letting the user choose them. In a multi-user system, this interest may not be given, and one could imagine the worst case of an "arms race" situation, where applications end up setting their priorities to the maximum value. If all applications do this, the end result is a fair allocation in which the priority mechanism is implicitly eliminated, and no major harm is done.

## 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.
- [RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140, April 1997.
- [RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.
- [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

### [rtcweb-rtp-usage]

Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-13.txt (work in progress), April 2014.

### [rtcweb-usecases]

Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-14.txt (work in progress), February 2014.

### [tech-report-fse]

Islam, S., Welzl, M., Gjessing, S., and N. Khademi, "Coupled Congestion Control for RTP Media", University of Oslo Department of Informatics technical report 440, available from <http://safiquili.at.ifi.uio.no/paper/fse-tech-report.pdf> , 2014.

### [transport-multiplex]

Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-07.txt (work in progress), October 2013.

## Appendix A. Example algorithm - Passive FSE

Active algorithms calculate the rates for all the flows in the FG and actively distribute them. In a passive algorithm, UPDATE returns a rate that should be used instead of the rate that the congestion controller has determined. This can make a passive algorithm easier to implement; however, the resulting dynamics are not fully understood. The algorithm described below is to be considered as highly experimental and did not perform as well as the active variants in simulations.

This passive version of the FSE stores the following information in addition to the variables described in Section 5.2:

- o The desired rate DR. This can be smaller than the calculated rate if the application feeding into the flow has less data to send than the congestion controller would allow. In case of a bulk transfer, DR must be set to CC\_R received from the flow's congestion module.

The passive version of the FSE contains one static variable per FG called TLO (Total Leftover Rate -- used to let a flow 'take' bandwidth from application-limited or terminated flows) which is initialized to 0. For the passive version, S\_CR is limited to increase or decrease as conservatively as a flow's congestion controller decides in order to prohibit sudden rate jumps.

- (1) When a flow *f* starts, it registers itself with SBD and the FSE. FSE\_R and DR are initialized with the congestion controller's initial rate. SBD will assign the correct FGI. When a flow is assigned an FGI, it adds its FSE\_R to S\_CR.
- (2) When a flow *f* stops, it sets its DR to 0 and sets P to -1.
- (3) Every time the congestion controller of the flow *f* determines a new sending rate CC\_R, assuming the flow's new desired rate new\_DR to be "infinity" in case of a bulk data transfer with an unknown maximum rate, the flow calls UPDATE, which carries out the tasks listed below to derive the flow's new sending rate, Rate. A flow's UPDATE function uses a few local (i.e. per-flow) temporary variables, which are all initialized to 0: DELTA, new\_S\_CR and S\_P.
  - (a) For all the flows in its FG (including itself), it calculates the sum of all the calculated rates, new\_S\_CR. Then it calculates the difference between FSE\_R(*f*) and CC\_R, DELTA.

```

for all flows i in FG do
    new_S_CR = new_S_CR + FSE_R(i)
end for
DELTA = CC_R - FSE_R(f)

```

- (b) It updates S\_CR, FSE\_R(*f*) and DR(*f*).

```

FSE_R(f) = CC_R
if DELTA > 0 then // the flow's rate has increased
    S_CR = S_CR + DELTA
else if DELTA < 0 then
    S_CR = new_S_CR + DELTA
end if
DR(f) = min(new_DR, FSE_R(f))

```

- (c) It calculates the leftover rate TLO, removes the terminated flows from the FSE and calculates the sum of all the priorities, S\_P.

```

for all flows i in FG do
  if P(i)<0 then
    delete flow
  else
    S_P = S_P + P(i)
  end if
end for
if DR(f) < FSE_R(f) then
  TLO = TLO + (P(f)/S_P) * S_CR - DR(f)
end if

```

- (d) It calculates the sending rate, Rate.

```

Rate = min(new_DR, (P(f)*S_CR)/S_P + TLO)

if Rate != new_DR and TLO > 0 then
  TLO = 0 // f has 'taken' TLO
end if

```

- (e) It updates DR(f) and FSE\_R(f) with Rate.

```

if Rate > DR(f) then
  DR(f) = Rate
end if
FSE_R(f) = Rate

```

The goals of the flow algorithm are to achieve prioritization, improve network utilization in the face of application-limited flows, and impose limits on the increase behavior such that the negative impact of multiple flows trying to increase their rate together is minimized. It does that by assigning a flow a sending rate that may not be what the flow's congestion controller expected. It therefore builds on the assumption that no significant inefficiencies arise from temporary application-limited behavior or from quickly jumping to a rate that is higher than the congestion controller intended. How problematic these issues really are depends on the controllers in use and requires careful per-controller experimentation. The coupled congestion control mechanism described here also does not require all controllers to be equal; effects of heterogeneous controllers, or homogeneous controllers being in different states, are also subject to experimentation.

This algorithm gives all the leftover rate of application-limited flows to the first flow that updates its sending rate, provided that this flow needs it all (otherwise, its own leftover rate can be taken by the next flow that updates its rate). Other policies could be applied, e.g. to divide the leftover rate of a flow equally among all other flows in the FGI.

#### A.1. Example operation (passive)

In order to illustrate the operation of the passive coupled congestion control algorithm, this section presents a toy example of two flows that use it. Let us assume that both flows traverse a common 10 Mbit/s bottleneck and use a simplistic congestion controller that starts out with 1 Mbit/s, increases its rate by 1 Mbit/s in the absence of congestion and decreases it by 2 Mbit/s in the presence of congestion. For simplicity, flows are assumed to always operate in a round-robin fashion. Rate numbers below without units are assumed to be in Mbit/s. For illustration purposes, the actual sending rate is also shown for every flow in FSE diagrams even though it is not really stored in the FSE.

Flow #1 begins. It is a bulk data transfer and considers itself to have top priority. This is the FSE after the flow algorithm's step 1:

#	FGI	P	FSE_R	DR	Rate
1	1	1	1	1	1

S\_CR = 1, TLO = 0

Its congestion controller gradually increases its rate. Eventually, at some point, the FSE should look like this:

#	FGI	P	FSE_R	DR	Rate
1	1	1	10	10	10

S\_CR = 10, TLO = 0

Now another flow joins. It is also a bulk data transfer, and has a lower priority (0.5):

#	FGI	P	FSE_R	DR	Rate
1	1	1	10	10	10
2	1	0.5	1	1	1

S\_CR = 11, TLO = 0

Now assume that the first flow updates its rate to 8, because the total sending rate of 11 exceeds the total capacity. Let us take a closer look at what happens in step 3 of the flow algorithm.

CC\_R = 8. new\_DR = infinity.

3 a) new\_S\_CR = 11; DELTA = 8 - 10 = -2.

3 b) FSE\_Rf) = 8. DELTA is negative, hence S\_CR = 9;  
DR(f) = 8.

3 c) S\_P = 1.5.

3 d) new sending rate =  $\min(\text{infinity}, 1/1.5 * 9 + 0) = 6$ .

3 e) FSE\_R(f) = 6.

The resulting FSE looks as follows:

#	FGI	P	FSE_R	DR	Rate
1	1	1	6	8	6
2	1	0.5	1	1	1

S\_CR = 9, TLO = 0

The effect is that flow #1 is sending with 6 Mbit/s instead of the 8 Mbit/s that the congestion controller derived. Let us now assume that flow #2 updates its rate. Its congestion controller detects that the network is not fully saturated (the actual total sending rate is 6+1=7) and increases its rate.

```

CC_R=2. new_DR = infinity.
3 a) new_S_CR = 7; DELTA = 2 - 1 = 1.
3 b) FSE_R(f) = 2. DELTA is positive, hence S_CR = 9 + 1 = 10;
    DR(f) = 2.
3 c) S_P = 1.5.
3 d) new sending rate = min(infinity, 0.5/1.5 * 10 + 0) = 3.33.
3 e) DR(f) = FSE_R(f) = 3.33.

```

The resulting FSE looks as follows:

#	FGI	P	FSE_R	DR	Rate
1	1	1	6	8	6
2	1	0.5	3.33	3.33	3.33

S\_CR = 10, TLO = 0

The effect is that flow #2 is now sending with 3.33 Mbit/s, which is close to half of the rate of flow #1 and leads to a total utilization of  $6(\#1) + 3.33(\#2) = 9.33$  Mbit/s. Flow #2's congestion controller has increased its rate faster than the controller actually expected. Now, flow #1 updates its rate. Its congestion controller detects that the network is not fully saturated and increases its rate. Additionally, the application feeding into flow #1 limits the flow's sending rate to at most 2 Mbit/s.

```

CC_R=7. new_DR=2.
3 a) new_S_CR = 9.33; DELTA = 1.
3 b) FSE_R(f) = 7, DELTA is positive, hence S_CR = 10 + 1 = 11;
    DR = min(2, 7) = 2.
3 c) S_P = 1.5; DR(f) < FSE_R(f), hence TLO = 1/1.5 * 11 - 2 = 5.33.
3 d) new sending rate = min(2, 1/1.5 * 11 + 5.33) = 2.
3 e) FSE_R(f) = 2.

```

The resulting FSE looks as follows:

#	FGI	P	FSE_R	DR	Rate
1	1	1	2	2	2
2	1	0.5	3.33	3.33	3.33

S\_CR = 11, TLO = 5.33

Now, the total rate of the two flows is  $2 + 3.33 = 5.33$  Mbit/s, i.e. the network is significantly underutilized due to the limitation of flow #1. Flow #2 updates its rate. Its congestion controller detects that the network is not fully saturated and increases its rate.

```
CC_R=4.33. new_DR = infinity.
3 a) new_S_CR = 5.33; DELTA = 1.
3 b) FSE_R(f) = 4.33. DELTA is positive, hence S_CR = 12;
    DR(f) = 4.33.
3 c) S_P = 1.5.
3 d) new sending rate: min(infinity, 0.5/1.5 * 12 + 5.33 ) = 9.33.
3 e) FSE_R(f) = 9.33, DR(f) = 9.33.
```

The resulting FSE looks as follows:

#	FGI	P	FSE_R	DR	Rate
1	1	1	2	2	2
2	1	0.5	9.33	9.33	9.33

S\_CR = 12, TLO = 0

Now, the total rate of the two flows is  $2 + 9.33 = 11.33$  Mbit/s. Finally, flow #1 terminates. It sets P to -1 and DR to 0. Let us assume that it terminated late enough for flow #2 to still experience the network in a congested state, i.e. flow #2 decreases its rate in the next iteration.

```
CC_R = 7.33. new_DR = infinity.
3 a) new_S_CR = 11.33; DELTA = -2.
3 b) FSE_R(f) = 7.33. DELTA is negative, hence S_CR = 9.33;
    DR(f) = 7.33.
3 c) Flow 1 has P = -1, hence it is deleted from the FSE.
    S_P = 0.5.
3 d) new sending rate: min(infinity, 0.5/0.5*9.33 + 0) = 9.33.
3 e) FSE_R(f) = DR(f) = 9.33.
```

The resulting FSE looks as follows:

#	FGI	P	FSE_R	DR	Rate
2	1	0.5	9.33	9.33	9.33

$S_{CR} = 9.33$ ,  $TLO = 0$

## Appendix B. Change log

### B.1. Changes from -00 to -01

- o Added change log.
- o Updated the example algorithm and its operation.

### B.2. Changes from -01 to -02

- o Included an active version of the algorithm which is simpler.
- o Replaced "greedy flow" with "bulk data transfer" and "non-greedy" with "application-limited".
- o Updated new\_CR to CC\_R, and CR to FSE\_R for better understanding.

### B.3. Changes from -02 to -03

- o Included an active conservative version of the algorithm which reduces queue growth and packet loss; added a reference to a technical report that shows these benefits with simulations.
- o Moved the passive variant of the algorithm to appendix.

## Authors' Addresses

Michael Welzl  
University of Oslo  
PO Box 1080 Blindern  
Oslo, N-0316  
Norway

Phone: +47 22 85 24 20  
Email: [michawe@ifi.uio.no](mailto:michawe@ifi.uio.no)

Safiqul Islam  
University of Oslo  
PO Box 1080 Blindern  
Oslo, N-0316  
Norway

Phone: +47 22 84 08 37  
Email: safiquli@ifi.uio.no

Stein Gjessing  
University of Oslo  
PO Box 1080 Blindern  
Oslo, N-0316  
Norway

Phone: +47 22 85 24 44  
Email: steing@ifi.uio.no



RMCAT WG  
Internet-Draft  
Intended status: Informational  
Expires: January 5, 2015

M. Zanaty  
Cisco  
V. Singh  
Aalto University  
S. Nandakumar  
Cisco  
Z. Sarker  
Ericsson AB  
July 4, 2014

RTP Application Interaction with Congestion Control  
draft-zanaty-rmcat-app-interaction-01

Abstract

Interactive real-time media applications that use the Real-time Transport Protocol (RTP) over the User Datagram Protocol (UDP) must use congestion control techniques above the UDP layer since it provides none. This memo describes the interactions and conceptual interfaces necessary between the application components that relate to congestion control, including the RTP layer, the higher-level media codec control layer, and the lower-level transport interface, as well as components dedicated to congestion control functions.

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 5, 2015.

Copyright Notice

Copyright (c) 2014 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 . . . . .	2
2. Key Words for Requirements . . . . .	3
3. Conceptual Model . . . . .	4
4. Implementation Model . . . . .	5
5. Interfaces and Interactions . . . . .	6
5.1. Config - Codec Interactions . . . . .	6
5.2. Config - RTP/RTCP Interactions . . . . .	6
5.3. Codec - RTP Interactions . . . . .	6
5.4. Codec - CC Interactions . . . . .	7
5.5. RTP - CC Interactions . . . . .	9
5.6. CC - UDP Interactions . . . . .	9
5.7. CC - Shared State Interactions . . . . .	10
6. Acknowledgements . . . . .	10
7. IANA Considerations . . . . .	10
8. Security Considerations . . . . .	10
9. References . . . . .	11
9.1. Normative References . . . . .	11
9.2. Informative References . . . . .	12
Authors' Addresses . . . . .	13

## 1. Introduction

Interactive real-time media applications most commonly use RTP [RFC3550] over UDP [RFC0768]. Since UDP provides no form of congestion control, which is essential for any application deployed on the Internet, these RTP applications have historically implemented one of the following options at the application layer to address their congestion control requirements.

1. For media with relatively low packet rates and bit rates, such as many speech codecs, some applications use a simple form of congestion control that stops transmission permanently or temporarily after observing significant packet loss over a significant period of time, similar to the RTP circuit breakers [I-D.ietf-avtcore-rtp-circuit-breakers].

2. Some applications have no explicit congestion control, despite the clear requirements in RTP and its profiles AVP [RFC3551] and AVPF [RFC4585], under the expectation that users will terminate media flows that are significantly impaired by congestion (in essence, human circuit breakers).
3. For media with substantially higher packet rates and bit rates, such as many video codecs, various non-standard congestion control techniques are often used to adapt transmission rate based on receiver feedback.
4. Some experimental applications use standardized techniques such as TCP-Friendly Rate Control (TFRC) [RFC5348]. However, for various reasons, these have not been widely deployed.

The RTP Media Congestion Avoidance Techniques (RMCAT) working group was chartered to standardize appropriate and effective congestion control for RTP applications. It is expected such applications will migrate from the above historical solutions to the RMCAT solution(s).

The RMCAT requirements [I-D.ietf-rmcat-cc-requirements] include low delay, reasonably high throughput, fast reaction to capacity changes including routing or interface changes, stability without over-reaction or oscillation, fair bandwidth sharing with other instances of itself and TCP flows, sharing information across multiple flows when possible [I-D.welzl-rmcat-coupled-cc], and performing as well or better in networks which support Active Queue Management (AQM), Explicit Congestion Notification (ECN), or Differentiated Services Code Points (DSCP).

In order to meet these requirements, interactions are necessary between the application's congestion controller, the RTP layer, media codecs, other components, and the OS UDP stack. This memo discusses these interactions, presents a conceptual model of the required interfaces based on a simplified application decomposition, and proposes specific information exchange across these interfaces along with corresponding component behavior.

Note that RTP can also operate over other transports with integrated congestion control such as TCP [RFC5681] and DCCP [RFC4340], but that is beyond the scope of RMCAT and this memo.

## 2. Key Words for Requirements

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 [RFC2119].

### 3. Conceptual Model

It is useful to decompose an RTP application into several components to facilitate understanding and discussion of where congestion control functions operate, and how they interface with the other components. The conceptual model in Figure 1 consists of the following components.

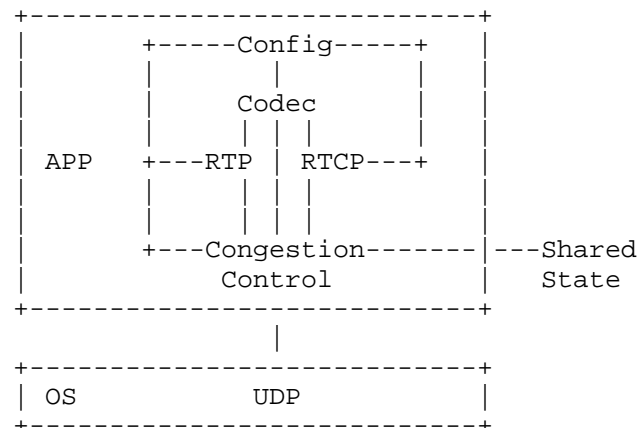


Figure 1

- o **APP**: Application containing one or more RTP streams and the corresponding media codecs and congestion controllers. For example, a WebRTC browser.
- o **Config**: Configuration specified by the application that provides the media and transport parameters, RTP and RTCP parameters and extensions, and congestion control parameters. For example, a WebRTC Javascript application may use the 'constraints' API to affect the media configuration, and SDP applications may negotiate the media and transport parameters with the remote peer. This determines the initial static configuration negotiated in session establishment. The dynamic state may differ due to congestion or other factors, but still must conform to limits established in the config.
- o **Codec**: Media encoder/decoder or other source/sink for the RTP payload. The codec may be, for example, a simple monaural audio format, a complex scalable video codec with several dependent layers, or a source/sink with no live encoding/decoding such as a mixer which selectively switches and forwards streams rather than mixes media.

- o RTP: Standard RTP stack functions, including media packetization / depacketization and header processing, but excluding existing extensions and possible new extensions specific to congestion control (CC) such as absolute timestamps or relative transmission time offsets in RTP header extensions. RTCP: Standard RTCP functions, including sender reports, receiver reports, extended reports, circuit breakers [I-D.ietf-avtcore-rtp-circuit-breakers], feedback messages such as NACK [RFC4585] and codec control messages such as TMMBR [RFC5104], but excluding existing extensions and possible new extensions specific to congestion control (CC) such as REMB [I-D.alvestrand-rmcat-remb] (for receiver-side CC), ACK (for sender-side CC), absolute and/or relative timestamps (for sender-side or receiver-side CC), etc.
- o Congestion Control: All functions directly responsible for congestion control, including possible new RTP/RTCP extensions, send rate computation (for sender-side CC), receive rate computation (for receiver-side CC), other statistics, and control of the UDP sockets including packet scheduling for traffic shaping/pacing.
- o Shared State: Storage and exchange of congestion control state for multiple flows within the application and beyond it.
- o OS: Operating System containing the UDP socket interface and other network functions such as ECN, DSCP, physical interface events, interface-level traffic shaping and packet scheduling, etc.

#### 4. Implementation Model

There are advantages and drawbacks to implementing congestion control in the application layer. It avoids OS dependencies and allows for rapid experimentation, evolution and optimization for each application. However, it also puts the burden on all applications, which raises the risks of improper or divergent implementations. One motivation of this memo is to mitigate such risks by giving proper guidance on how the application components relating to congestion control should interact.

Another drawback of congestion control in the application layer is that any decomposition, including the one presented in Figure 1, is purely conceptual and illustrative, since implementations have differing designs and decompositions. Conversely, this can be viewed as an advantage to distribute congestion control functions wherever expedient without rigid interfaces. For example, they may be distributed within the RTP/RTCP stack itself, so the separate components in Figure 1 are combined into a single RTP+RTCP+CC component as shown in Figure 2.

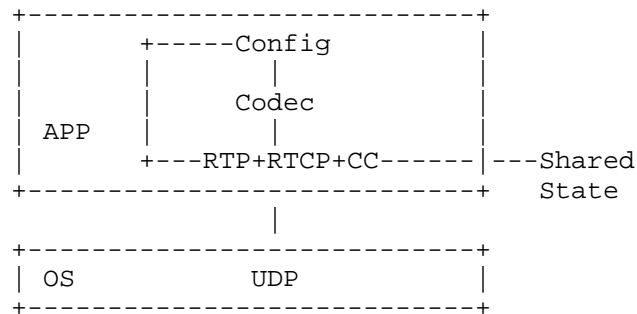


Figure 2

The conceptual model in Figure 1 will be used throughout this memo to establish clearer boundaries between functions. But actual implementations may be closer to the looser model in [Singh12].

## 5. Interfaces and Interactions

### 5.1. Config - Codec Interactions

The primary interactions between the config and the codec that are relevant to congestion control are the multiplexing of media streams [I-D.ietf-mmusic-sdp-bundle-negotiation] and RTP/RTCP [RFC5761] on the same UDP port.

The config also establishes limits for the codec such as maximum bit rate and other codec-specific parameters. For example, a video codec config often sets limits on maximum resolution and frame rate as well as bit rate.

### 5.2. Config - RTP/RTCP Interactions

The config establishes the negotiated RTP and RTCP attributes and extensions such as Extended Reports (XR), reduced size [RFC5506], codec control [RFC5104], transmission time [RFC5450], etc.

### 5.3. Codec - RTP Interactions

Packetization of codec frames into RTP packets can be an important interaction. Some network interfaces may benefit from small packet sizes well below the MTU, while others may benefit from large packets approaching the MTU. Equalizing packet sizes of a frame may also be beneficial in some cases, rather than a combination of large and small packets. For example, in some FEC schemes, the FEC bandwidth overhead depends on the largest source packet size. Equalizing the

source packet sizes can yield lower overhead than a combination of large and small packets.

#### 5.4. Codec - CC Interactions

**Allowed Rate (from CC to Codec):** The max transmit rate allowed over the next time interval. The time interval may be specified or may use a default, for example, one second. The rate may be specified in bytes or packets or both. The rate must never exceed permanent limits established in session signaling such as the SDP bandwidth attribute [RFC4566] nor temporary limits in RTCP such as TMMBR [RFC5104] or REMB [I-D.alvestrand-rmcat-remb]. This is the most important interface among all components, and is always required in any RMCAT solution. In the simplest possible solution, it may be the only CC interface required.

**Media Elasticity (from Codec to CC):** Many live media encoders are highly elastic, often able to achieve any target bit rate within a wide range, by adapting the media quality. For example, a video encoder may support any bit rate within a range of a few tens or hundreds of kbps up to several Mbps, with rate changes registering as fast as the next video frame, although there may be limitations in the frequency of changes. Other encoders may be less elastic, supporting a narrower rate range, coarser granularity of rate steps, slower reaction to rate changes, etc. Other media, particularly some audio codecs, may be fully inelastic with a single fixed rate. CC can beneficially use codec elasticity, if provided, to plan Allowed Rate changes, especially when there are multiple flows sharing CC state and bandwidth.

**Startup Ramp (from Codec to CC, and from CC to Codec):** Startup is an important moment in a conversation. Rapid rate adaptation during startup is therefore important. The codec should minimize its startup media rate as much as possible without adversely impacting the user experience, and support a strategy for rapid rate ramp. The CC should allow the highest startup media rate as possible without adversely impacting network conditions, and also support rapid rate ramp until stabilizing on the available bandwidth. Startup can be viewed as a negotiation between the codec and the CC. The codec requests a startup rate and ramp, and the CC responds with the allowable parameters which may be lower/slower. The RMCAT requirements also include the possibility of bandwidth history to further accelerate or even eliminate startup ramp time. While this is highly desirable from an application viewpoint, it may be less acceptable to network operators, since it is in essence a gamble on current congestion state matching historical state, with the potential for significant congestion contribution if the gamble was

wrong. Note that startup can often commence before user interaction or conversation to reduce the chance of clipped media.

Delay Tolerance (from Codec to CC): An ideal CC will always minimize delay and target zero. However, real solutions often need a real non-zero delay tolerance. The codec should provide an absolute delay tolerance, perhaps expressed as an impairment factor to mix with other metrics.

Loss Tolerance (from Codec to CC): An ideal CC will always minimize packet loss and target zero. However, real solutions often need a real non-zero loss tolerance. The codec should provide an absolute loss tolerance, perhaps expressed as an impairment factor to mix with other metrics. Note this is unrecoverable post-repair loss after retransmission or forward error correction.

Throughput Sensitivity (from Codec to CC): An ideal CC will always maximize throughput. However, real solutions often need a trade-off between throughput and other metrics such as delay or loss. The codec should provide throughput sensitivity, perhaps expressed as an impairment factor (for low throughputs) to mix with other metrics.

Rate Stability (from Codec to CC): The CC algorithm must strike a balance between rate stability and fast reaction to changes in available bandwidth. The codec should provide its preference for rate stability versus fast and frequent reaction to rate changes, perhaps expressed as an impairment factor (for high rate variance over short timescales) to mix with other metrics.

Forward Error Correction (FEC): Simple FEC schemes like XOR Parity codes [RFC5109] may not handle consecutive or burst loss well. More complex FEC schemes like Reed-Solomon [RFC6865] or Raptor [RFC6330] codes are more effective at handling bursty loss. The sensitivity to packet loss therefore depends on the media (source) encoding as well as the FEC (channel) encoding, and this sensitivity may differ for different loss patterns like random, periodic, or consecutive loss. Expressing this sensitivity to the congestion controller may help it choose the right balance between optimizing for throughput versus low loss.

Probing for Available Bandwidth: FEC can also be used to probe for additional available bandwidth, if the application desires a higher target rate than the current rate. FEC is preferable to synthetic probes since any contribution to congestion by the FEC probe will not impact the post-repair loss rate of the source media flow while synthetic probes may adversely affect the loss rate [Nagy14]. Note that any use of FEC or retransmission must ensure that the total flow

of all packets including FEC, retransmission and original media never exceeds the Allowed Rate.

#### 5.5. RTP - CC Interactions

**RTP Circuit Breakers:** The intent behind RTP circuit breakers [I-D.ietf-avtcore-rtp-circuit-breakers] is to provide a kill switch of last resort, not true congestion control. The breakers should never trip when an effective congestion control is operating. This may impose some boundaries on RMCAT solutions to ensure the congestion control never approaches situations which may trigger the breakers.

**RTCP Feedback:** The primary method of communicating CC information is RTCP.

**RTP Header Extensions:** While RTCP is likely to be the primary carrier of CC feedback, the RMCAT requirements also include the possibility of using RTP header extensions in bidirectional flows for CC feedback. Transmission time [RFC5450], or possibly absolute time, also use header extensions, as would any per packet priority markings expected to survive across different networks and administrative domains.

#### 5.6. CC - UDP Interactions

**Pacing / Shaping:** Simple pacing / shaping strategies delay the transmission of packets to equalize inter-packet time intervals, assuming the bottleneck is most sensitive to packet rate. More complex pacing strategies may go beyond simple even distribution of transmission times. For example, Sprout [Winstein13] attempts to predict the optimal transmission time (and rate) with the highest probability of success for each packet based on channel statistics. Pacing may be always on, or adaptively enabled / disabled based on congestion state to minimize delay. Pacing may be performed within the CC for a single flow or across multiple flows. It may also be performed across all or selective traffic over the network interface if the OS supports interface-level traffic shaping.

**Detection of Transport Capabilities:** The CC can query the OS for useful transport capabilities such as ECN, DSCP, traffic shaping, etc. This may also aid upper layers in making better decisions such as whether or not to multiplex media streams. For example, if audio can be given differentiated network treatment from video when using separate ports.

ECN: If the OS and transport path support ECN, the CC can react faster than a loss-based CC and more reliably to congestion onset and abatement.

DSCP: If the OS and transport path support DSCP, the CC can map per-packet priority from RTP header extensions to DSCP (and layer 2 QoS if available) for better network handling under congestion.

AQM: If AQM is present in the bottleneck, and working effectively, there should be little or no excess delay observed when varying the transmission rate. The loss of such delay signals may hinder the performance of congestion control algorithms that are highly dependent on delay variation for adapting transmission rate. If the application has knowledge of the presence of AQM, through any means which are beyond the scope of this memo, it should communicate this to the CC. The CC may use this to alter its signal collection and rate adaptation strategies. The CC must not rely solely on this as a reliable indicator. It must continue to monitor statistics to validate this application hint, and react appropriately if the statistics suggest different network behavior.

#### 5.7. CC - Shared State Interactions

Multiple Flows: Sharing state across multiple flows within the application can yield better CC decisions. Sharing state across even more flows beyond the application can yield even better CC decisions. The actual benefits and mechanisms of state sharing and coupled CC are described in [I-D.welzl-rmcat-coupled-cc].

Weighted Fairness: An important consideration in CC of multiple flows is their relative application-specified weights. Within an application, it is likely the different flows have different rate requirements, so equal bandwidth sharing may not be fair nor desirable, and weighted fairness may be required.

#### 6. Acknowledgements

The RMCAT design team discussions contributed to this memo.

#### 7. IANA Considerations

This memo includes no request to IANA.

#### 8. Security Considerations

Amplification attacks often use UDP traffic to launch denial of service attacks. Attackers may attempt to subvert congestion control protocols in UDP applications to launch amplification attacks by

signaling more bandwidth than is actually available. For example, sending a victim a forged REMB or a few fast ACKs may result in the victim sending a high rate RTP stream. Attacks on conference servers could lead to further amplification if it distributes the streams to many others. One mitigation is to use SRTCP for congestion control messages where supported. Even if SRTCP is only authenticated not encrypted, SRTCP packets should always pass authentication checks before any message contents are interpreted. Non-secure RTCP should be avoided where possible.

## 9. References

### 9.1. Normative References

- [I-D.alvestrand-rmcat-remb]  
Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", draft-alvestrand-rmcat-remb-03 (work in progress), October 2013.
- [I-D.ietf-avtcore-rtp-circuit-breakers]  
Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", draft-ietf-avtcore-rtp-circuit-breakers-05 (work in progress), February 2014.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]  
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-07 (work in progress), April 2014.
- [I-D.ietf-rmcat-cc-requirements]  
Jesup, R., "Congestion Control Requirements For RMCAT", draft-ietf-rmcat-cc-requirements-04 (work in progress), April 2014.
- [I-D.ietf-rmcat-eval-criteria]  
Singh, V. and J. Ott, "Evaluating Congestion Control for Interactive Real-time Media", draft-ietf-rmcat-eval-criteria-01 (work in progress), March 2014.
- [I-D.welzl-rmcat-coupled-cc]  
Welzl, M., Islam, S., and S. Gjessing, "Coupled congestion control for RTP media", draft-welzl-rmcat-coupled-cc-03 (work in progress), May 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [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.
- [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.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

## 9.2. Informative References

- [Nagy14] Nagy, M., Singh, V., Ott, J., and L. Eggert, "Congestion Control using FEC for Conversational Multimedia Communication", Proc. of 5th ACM International Conference on Multimedia Systems , 3 2014.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.

- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, August 2011.
- [RFC6865] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6865, February 2013.
- [Singh12] Singh, V., Ott, J., and C. Perkins, "Congestion Control for Interactive Media: Control Loops & APIs", Proc. of IAB/IRTF Workshop on Congestion Control for Interactive RTC , 7 2012.
- [Winstein13] Winstein,, K., Sivaraman,, A., and H. Balakrishnan, "Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks", Proc. of the 10th USENIX Symposium on Networked Systems Design and Implementation , 4 2013.

## Authors' Addresses

Mo Zanaty  
Cisco  
Raleigh, NC  
USA

Email: mzanaty@cisco.com

Varun Singh  
Aalto University  
Espoo, FIN  
Finland

Email: varun@comnet.tkk.fi

Suhas Nandakumar  
Cisco  
San Jose, CA  
USA

Email: [snandaku@cisco.com](mailto:snandaku@cisco.com)

Zaheduzzaman Sarker  
Ericsson AB  
Luleae  
Sweden

Email: [zaheduzzaman.sarker@ericsson.com](mailto:zaheduzzaman.sarker@ericsson.com)

Network Working Group  
Internet Draft  
Intended Status: Informational  
Expires: September 14, 2014

X. Zhu  
R. Pan  
Cisco Systems  
March 13, 2014

NADA: A Unified Congestion Control Scheme for Real-Time Media  
draft-zhu-rmcat-nada-03

Abstract

This document describes a scheme named network-assisted dynamic adaptation (NADA), a novel congestion control approach for interactive real-time media applications, such as video conferencing. In the proposed scheme, the sender regulates its sending rate based on either implicit or explicit congestion signaling, in a unified approach. The scheme can benefit from explicit congestion notification (ECN) markings from network nodes. It also maintains consistent sender behavior in the absence of such markings, by reacting to queuing delays and packet losses instead.

We present here the overall system architecture, recommended behaviors at the sender and the receiver, as well as expected network node operations. Results from extensive simulation studies of the proposed scheme are available upon request.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

## Copyright and License 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. Terminology . . . . .	3
3. System Model . . . . .	3
4. Network Node Operations . . . . .	4
4.1 Default behavior of drop tail . . . . .	4
4.2 ECN marking . . . . .	4
4.3 PCN marking . . . . .	5
4.4 Comments and Discussions . . . . .	6
5. Receiver Behavior . . . . .	6
5.1 Monitoring per-packet statistics . . . . .	6
5.2 Calculating time-smoothed values . . . . .	7
5.3 Sending periodic feedback . . . . .	7
5.4 Discussions on one-way delay measurements . . . . .	7
6. Sender Behavior . . . . .	8
6.1 Video encoder rate control . . . . .	9
6.2 Rate shaping buffer . . . . .	9
6.3 Reference rate calculator . . . . .	9
6.4 Video target rate and sending rate calculator . . . . .	11
6.5 Slow-start behavior . . . . .	11
7. Incremental Deployment . . . . .	12
8. Implementation Status . . . . .	12
9. IANA Considerations . . . . .	12
10. References . . . . .	12
10.1 Normative References . . . . .	12
10.2 Informative References . . . . .	12
Authors' Addresses . . . . .	13

## 1. Introduction

Interactive real-time media applications introduce a unique set of challenges for congestion control. Unlike TCP, the mechanism used for real-time media needs to adapt fast to instantaneous bandwidth changes, accommodate fluctuations in the output of video encoder rate control, and cause low queuing delay over the network. An ideal scheme should also make effective use of all types of congestion signals, including packet losses, queuing delay, and explicit congestion notification (ECN) markings.

Based on the above considerations, we present a scheme named network-assisted dynamic adaptation (NADA). The proposed design benefits from explicit congestion control signals (e.g., ECN markings) from the network, and remains compatible in the presence of implicit signals (delay or loss) only. In addition, it supports weighted bandwidth sharing among competing video flows.

This documentation describes the overall system architecture, recommended designs at the sender and receiver, as well as expected network nodes operations. The signaling mechanism consists of standard RTP timestamp [RFC3550] and standard RTCP feedback reports.

## 2. Terminology

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].

## 3. System Model

The system consists of the following elements:

- \* Incoming media stream, in the form of consecutive raw video frames and audio samples;
- \* Media encoder with rate control capabilities. It takes the incoming media stream and encodes it to an RTP stream at a target bit rate  $R_v$ . Note that the actual output rate from the encoder  $R_o$  may fluctuate randomly around the target  $R_v$ . Also, the encoder can only change its rate at rather coarse time intervals, e.g., once every 0.5 seconds.
- \* RTP sender, responsible for calculating the target bit rate  $R_n$  based on network congestion signals (delay or ECN marking reports from the receiver), and for regulating the actual sending rate  $R_s$  accordingly. A rate shaping buffer is employed

to absorb the instantaneous difference between video encoder output rate  $R_v$  and sending rate  $R_s$ . The buffer size  $L_s$ , together with  $R_n$ , influences the calculation of actual sending rate  $R_s$  and video encoder target rate  $R_v$ . The RTP sender also generates RTP timestamp in outgoing packets.

\* RTP receiver, responsible for measuring and estimating end-to-end delay  $d$  based on sender RTP timestamp. In the presence of packet losses and ECN markings, it also records the individual loss and marking events, and calculates the equivalent delay  $d_{\text{tilde}}$  that accounts for queuing delay, ECN marking, and packet losses. The receiver feeds such statistics back to the sender via periodic RTCP reports.

\* Network node, with several modes of operation. The system can work with the default behavior of a simple drop tail queue. It can also benefit from advanced AQM features such as RED-based ECN marking, and PCN marking using a token bucket algorithm.

In the following, we will elaborate on the respective operations at the network node, the receiver, and the sender.

#### 4. Network Node Operations

We consider three variations of queue management behavior at the network node, leading to either implicit or explicit congestion signals.

##### 4.1 Default behavior of drop tail

In conventional network with drop tail or RED queues, congestion is inferred from the estimation of end-to-end delay. No special action is required at network node.

Packet drops at the queue are detected at the receiver, and contributes to the calculation of the equivalent delay  $d_{\text{tilde}}$ .

##### 4.2 ECN marking

In this mode, the network node randomly marks the ECN field in the IP packet header following the Random Early Detection (RED) algorithm [RFC2309]. Calculation of the marking probability involves the following steps:

\* upon packet arrival, update smoothed queue size  $q_{\text{avg}}$  as:

$$q_{\text{avg}} = \alpha * q + (1 - \alpha) * q_{\text{avg}}.$$

The smoothing parameter  $\alpha$  is a value between 0 and 1. A value of

$\alpha=1$  corresponds to performing no smoothing at all.

\* calculate marking probability  $p$  as:

$p = 0$ , if  $q < q_{lo}$ ;

$p = p_{max} \frac{q_{avg} - q_{lo}}{q_{hi} - q_{lo}}$ , if  $q_{lo} \leq q < q_{hi}$ ;

$p = 1$ , if  $q \geq q_{hi}$ .

Here,  $q_{lo}$  and  $q_{hi}$  corresponds to the low and high thresholds of queue occupancy. The maximum parking probability is  $p_{max}$ .

The ECN markings events will contribute to the calculation of an equivalent delay  $d_{tilde}$  at the receiver. No changes are required at the sender.

#### 4.3 PCN marking

As a more advanced feature, we also envision network nodes which support PCN marking based on virtual queues. In such a case, the marking probability of the ECN bit in the IP packet header is calculated as follows:

\* upon packet arrival, meter packet against token bucket  $(r, b)$ ;

\* update token level  $b_{tk}$ ;

\* calculate the marking probability as:

$p = 0$ , if  $b - b_{tk} < b_{lo}$ ;

$p = p_{max} \frac{b - b_{tk} - b_{lo}}{b_{hi} - b_{lo}}$ , if  $b_{lo} \leq b - b_{tk} < b_{hi}$ ;

$p = 1$ , if  $b - b_{tk} \geq b_{hi}$ .

Here, the token bucket lower and upper limits are denoted by  $b_{lo}$  and  $b_{hi}$ , respectively. The parameter  $b$  indicates the size of the token bucket. The parameter  $r$  is chosen as  $r = \gamma C$ , where  $\gamma < 1$  is the target utilization ratio and  $C$  designates link capacity. The maximum marking probability is  $p_{max}$ .

The ECN markings events will contribute to the calculation of an equivalent delay  $d_{tilde}$  at the receiver. No changes are required at the

sender. The virtual queuing mechanism from the PCN marking algorithm will lead to additional benefits such as zero standing queues.

#### 4.4 Comments and Discussions

In all three flavors described above, the network queue operates with the simple first-in-first-out (FIFO) principle. There is no need to maintain per-flow state. Such a simple design ensures that the system can scale easily with large number of video flows and high link capacity.

The sender behavior stays the same in the presence of all types of congestion signals: delay, loss, ECN marking due to either RED/ECN or PCN algorithms. This unified approach allows a graceful transition of the scheme as the level of congestion in the network shifts dynamically between different regimes.

#### 5. Receiver Behavior

The role of the receiver is fairly straightforward. It is in charge of four steps: a) monitoring end-to-end delay/loss/marketing statistics on a per-packet basis; b) aggregating all forms of congestion signals in terms of the equivalent delay; c) calculating time-smoothed value of the congestion signal; and d) sending periodic reports back to the sender.

##### 5.1 Monitoring per-packet statistics

The receiver observes and estimates one-way delay  $d_n$  for the  $n$ -th packet, ECN marking event  $l_M$ , and packet loss event  $l_L$ . Here,  $l_M$  and  $l_L$  are binary indicators: the value of 1 corresponding to a marked or lost packet and value of 0 indicates no marking or loss.

The equivalent delay  $d_{\text{tilde}}$  is calculated as follows:

$$d_{\text{tilde}} = d_n + l_M d_M + l_L d_L,$$

where  $d_M$  is a prescribed fictitious delay value corresponding to the ECN marking event (e.g.,  $d_M = 200$  ms), and  $d_L$  is a prescribed fictitious delay value corresponding to the packet loss event (e.g.,  $d_L = 1$  second). By introducing a large fictitious delay penalty for ECN marking and packet losses, the proposed scheme leads to low end-to-end actual delays in the presence of such events.

While the value of  $d_M$  and  $d_L$  are fixed and predetermined in our current design, we also plan to investigate a scheme for automatically tuning these values based on desired bandwidth sharing behavior in the presence of other competing loss-based flows (e.g., loss-based TCP).

## 5.2 Calculating time-smoothed values

The receiver smoothes its observations via exponential averaging:

$$x_n = \alpha * d\_tilde + (1-\alpha) * x_n.$$

The weighting parameter  $\alpha$  adjusts the level of smoothing.

## 5.3 Sending periodic feedback

Periodically, the receiver sends back the updated value of  $x$  in RTCP messages, to aid the sender in its calculation of target rate. The size of acknowledgement packets are typically on the order of tens of bytes, and are significantly smaller than average video packet sizes. Therefore, the bandwidth overhead of the receiver acknowledgement stream is sufficiently low.

## 5.4 Discussions on one-way delay measurements

At the current stage, our proposed scheme relies on one-way delay (OWD) as the primary form of congestion indication. This implicitly relies on well-synchronized sender and receiver clocks, e.g., due to the presence of an auxiliary clock synchronization process. For deployment in the open Internet, however, this assumption may not always hold.

There are several ways to get around the clock synchronization issue by slightly tweaking the current design. One option is to work with relative OWD instead, by maintaining the minimum value of observed OWD over a longer time horizon and subtract that out from the observed absolute OWD value. Such an approach cancels out the fixed clock difference from the sender and receiver clocks, and has been widely adopted by other delay-based congestion control approaches such as LEDBAT [RFC6817]. As discussed in [RFC6817], the time horizon for tracking the minimum OWD needs to be chosen with care: long enough for an opportunity to observe the minimum OWD with zero queuing delay along the path, and sufficiently short so as to timely reflect "true" changes in minimum OWD introduced by route changes and other rare events.

Alternatively, one could move the per-packet statistical handling to the sender instead, and use RTT in lieu of OWD, assuming the per-packet ACKs are present. The main drawback of this latter approach, on the other hand, is that the scheme will be confused by congestion in the reverse direction.

Note that either approach involves no change in the proposed rate adaptation algorithm at the sender. Therefore, comparing the pros and cons regarding which delay metric to use can be kept as an orthogonal direction of investigation.

## 6. Sender Behavior

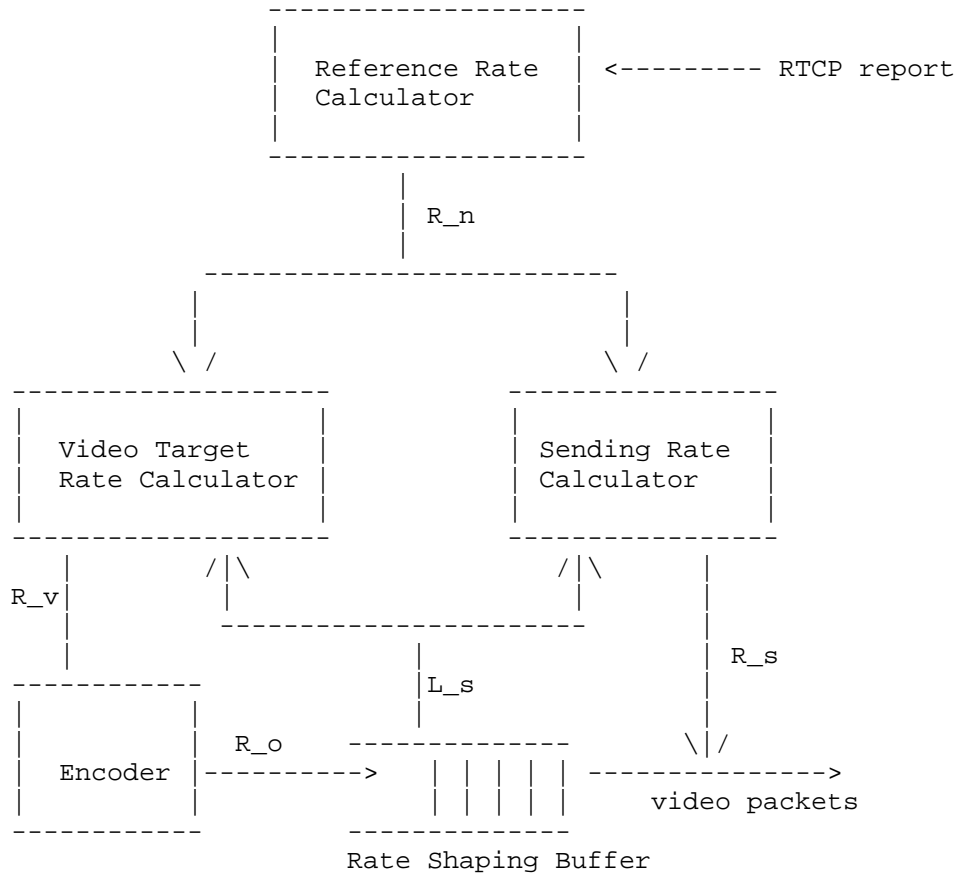


Figure 1 NADA Sender Structure

Figure 1 provides a more detailed view of the NADA sender. Upon receipt of an RTCP report from the receiver, the NADA sender updates its calculation of the reference rate  $R_n$  as a function of the network congestion signal. It further adjusts both the target rate for the live video encoder  $R_v$  and the sending rate  $R_s$  over the network based on the updated value of  $R_n$ , as well as the size of the rate shaping buffer.

The following sections describe these modules in further details, and explain how they interact with each other.

### 6.1 Video encoder rate control

The video encoder rate control procedure has the following characteristics:

- \* Rate changes can happen only at large intervals, on the order of seconds.
- \* Given a target rate  $R_o$ , the encoder output rate may randomly fluctuate around it.
- \* The encoder output rate is further constrained by video content complexity. The range of the final rate output is  $[R_{min}, R_{max}]$ . Note that it's content-dependent, and may change over time.

Note that operation of the live video encoder is out of the scope of our design for a congestion control scheme in NADA. Instead, its behavior is treated as a black box.

### 6.2 Rate shaping buffer

A rate shaping buffer is employed to absorb any instantaneous mismatch between encoder rate output  $R_o$  and regulated sending rate  $R_s$ . The size of the buffer evolves from time  $t-\tau$  to time  $t$  as:

$$L_s(t) = \max [0, L_s(t-\tau) + R_v \cdot \tau - R_s \cdot \tau].$$

A large rate shaping buffer contributes to higher end-to-end delay, which may harm the performance of real-time media communications. Therefore, the sender has a strong incentive to constrain the size of the shaping buffer. It can either deplete it faster by increasing the sending rate  $R_s$ , or limit its growth by reducing the target rate for the video encoder rate control  $R_v$ .

### 6.3 Reference rate calculator

The sender calculates the reference rate  $R_n$  based on network congestion information from receiver RTCP reports. It first compensates the effect of delayed observation by one round-trip time (RTT) via a linear predictor:

$$x\_hat = x\_n + \frac{x\_n - x\_n-1}{\delta} * \tau\_o \quad (1)$$

In (1), the arrival interval between the (n-1)-th the n-th packets is designated by  $\delta$ . The parameter  $\tau\_o$  is pre-configured to a fixed value. Typically, its value is comparable to the RTT experienced by the flow, but does not needs to be an exact match. Throughout all our simulation evaluations (see [Zhu-PV13]), we have been using the same fixed value of  $\tau\_o = 200\text{ms}$ .

The reference rate is then calculated as:

$$R\_n = R\_min + w * \frac{R\_max - R\_min}{x\_hat} * x\_ref \quad (2)$$

Here,  $R\_min$  and  $R\_max$  denote the content-dependent rate range the encoder can produce. The weight of priority level is  $w$ . The reference congestion signal  $x\_ref$  is chosen so that the maximum rate of  $R\_max$  can be achieved when  $x\_hat = w * x\_ref$ . The final target rate  $R\_n$  is clipped within the range of  $[R\_min, R\_max]$ .

The rationale of choose  $x\_ref$  to be the value of absolute one-way delay (i.e., only propagation delay along the path) is that ideally, we would want the video stream to reach the highest possible rate when the queue stays is empty, e.g., when bottleneck link rate exceeds  $R\_max$  of video. In practice, the stream can simply set  $x\_ref$  to be the minimum value of OWD observed over a long time horizon. Note also that the combination of  $w$  and  $x\_ref$  determines how sensitive the rate adaptation scheme is in reaction to fluctuations in observed signal  $x$ .

The sender does not need any explicit knowledge of the management scheme inside the network. Rather, it reacts to the aggregation of all forms of congestion indications (delay, loss, and marking) via the composite congestion signal  $x\_n$  from the receiver in a coherent manner.

#### 6.4 Video target rate and sending rate calculator

The target rate for the live video encoder is updated based on both the reference rate  $R_n$  and the rate shaping buffer size  $L_s$ , as follows:

$$R_v = R_o - \beta_v * \frac{L_s}{\tau_v}. \quad (3)$$

Similarly, the outgoing rate is regulated based on both the reference rate  $R_n$  and the rate shaping buffer size  $L_s$ , such that:

$$R_s = R_o + \beta_s * \frac{L_s}{\tau_v}. \quad (4)$$

In (3) and (4), the first term indicates the rate calculated from network congestion feedback alone. The second term indicates the influence of the rate shaping buffer. A large rate shaping buffer nudges the encoder target rate slightly below -- and the sending rate slightly above -- the reference rate  $R_n$ . Intuitively, the amount of extra rate offset needed to completely drain the rate shaping buffer within the same time frame of encoder rate adaptation  $\tau_v$  is given by  $L_s/\tau_v$ . The scaling parameters  $\beta_v$  and  $\beta_s$  can be tuned to balance between the competing goals of maintaining a small rate shaping buffer and deviating the system from the reference rate point.

#### 6.5 Slow-start behavior

Finally, special care needs to be taken during the startup phase of a video stream, since it may take several roundtrip-times before the sender can collect statistically robust information on network congestion. We propose to regulate the reference rate  $R_n$  to grow linearly in the beginning, no more than:  $R_{ss}$  at time  $t$ :

$$R_{ss}(t) = R_{min} + \frac{t-t_0}{T} (R_{max}-R_{min}).$$

The start time of the stream is  $t_0$ , and  $T$  represents the time horizon over which the slow-start mechanism is effective. The encoder target rate is chosen to be the minimum of  $R_n$  and  $R_{ss}$  during the first  $T$  seconds.

## 7. Incremental Deployment

One nice property of proposed design is the consistent video end point behavior irrespective of network node variations. This facilitates gradual, incremental adoption of the scheme.

To start off with, the proposed encoder congestion control mechanism can be implemented without any explicit support from the network, and rely solely on observed one-way delay measurements and packet loss ratios as implicit congestion signals.

When ECN is enabled at the network nodes with RED-based marking, the receiver can fold its observations of ECN markings into the calculation of the equivalent delay. The sender can react to these explicit congestion signals without any modification.

Ultimately, networks equipped with proactive marking based on token bucket level metering can reap the additional benefits of zero standing queues and lower end-to-end delay and work seamlessly with existing senders and receivers.

## 8. Implementation Status

The proposed NADA scheme has been implemented in the ns-2 simulation platform [ns2]. Extensive simulation evaluations of the scheme are documented in [Zhu-PV13].

The scheme has also been implemented in Linux. Initial set of testbed evaluation results have focused on the case of a single NADA flow over a single low-delay bottleneck link. More investigations are underway.

## 9. IANA Considerations

There are no actions for IANA.

## 10. References

### 10.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

### 10.2 Informative References

- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.
- [RFC6187] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, December 2012
- [ns2] "The Network Simulator - ns-2", <http://www.isi.edu/nsnam/ns/>
- [Zhu-PV13] Zhu, X. and Pan, R., "NADA: A Unified Congestion Control Scheme for Low-Latency Interactive Video", in Proc. IEEE International Packet Video Workshop (PV'13). San Jose, CA, USA. December 2013.

#### Authors' Addresses

Xiaoqing Zhu  
Cisco Systems,  
510 McCarthy Blvd,  
Milpitas, CA 95134, USA  
EMail: xiaoqzhu@cisco.com

Rong Pan  
Cisco Systems  
510 McCarthy Blvd,  
Milpitas, CA 95134, USA  
Email: ropan@cisco.com

Network Working Group  
Internet Draft  
Intended Status: Informational  
Expires: September 27, 2015

X. Zhu, R. Pan  
M. A. Ramalho, S. Mena  
C. Ganzhorn, P. E. Jones  
Cisco Systems  
S. De Aronco  
Ecole Polytechnique Federale de Lausanne  
March 26, 2015

NADA: A Unified Congestion Control Scheme for Real-Time Media  
draft-zhu-rmcat-nada-06

Abstract

Network-Assisted Dynamic Adaptation (NADA) is a novel congestion control scheme for interactive real-time media applications, such as video conferencing. In NADA, the sender regulates its sending rate based on either implicit or explicit congestion signaling in a consistent manner. As one example of explicit signaling, NADA can benefit from explicit congestion notification (ECN) markings from network nodes. It also maintains consistent sender behavior in the absence of explicit signaling by reacting to queuing delay and packet loss.

This document describes the overall system architecture for NADA, as well as recommended behavior at the sender and the receiver.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

## Copyright and License 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. Terminology . . . . .	3
3. System Model . . . . .	3
4. NADA Receiver Behavior . . . . .	4
4.1 Estimation of one-way delay and queuing delay . . . . .	4
4.2 Estimation of packet loss/marketing ratio . . . . .	5
4.3 Non-linear warping of delay . . . . .	6
4.4 Aggregating congestion signals . . . . .	7
4.5 Estimating receiving rate . . . . .	7
4.6 Sending periodic feedback . . . . .	7
4.7 Discussions on delay metrics . . . . .	8
5. NADA Sender Behavior . . . . .	9
5.1 Reference rate calculation . . . . .	10
5.1.1 Accelerated ramp up . . . . .	10
5.1.2. Gradual rate update . . . . .	11
5.2 Video encoder rate control . . . . .	12
5.3 Rate shaping buffer . . . . .	12
5.4 Adjusting video target rate and sending rate . . . . .	12
6. Incremental Deployment . . . . .	13
7. Implementation Status . . . . .	13
8. IANA Considerations . . . . .	14
9. References . . . . .	14
9.1 Normative References . . . . .	14
9.2 Informative References . . . . .	14
Appendix A. Network Node Operations . . . . .	15
A.1 Default behavior of drop tail . . . . .	16
A.2 ECN marking . . . . .	16
A.3 PCN marking . . . . .	16
Authors' Addresses . . . . .	17

## 1. Introduction

Interactive real-time media applications introduce a unique set of challenges for congestion control. Unlike TCP, the mechanism used for real-time media needs to adapt quickly to instantaneous bandwidth changes, accommodate fluctuations in the output of video encoder rate control, and cause low queuing delay over the network. An ideal scheme should also make effective use of all types of congestion signals, including packet loss, queuing delay, and explicit congestion notification (ECN) [RFC3168] markings.

Based on the above considerations, this document describes a scheme called network-assisted dynamic adaptation (NADA). The NADA design benefits from explicit congestion control signals (e.g., ECN markings) from the network, yet also operates when only implicit congestion indicators (delay and/or loss) are available. In addition, it supports weighted bandwidth sharing among competing video flows.

This documentation describes the overall system architecture, recommended designs at the sender and receiver, as well as expected network node operations. The signaling mechanism consists of standard RTP timestamp [RFC3550] and standard RTCP feedback reports.

## 2. Terminology

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].

## 3. System Model

The overall system consists of the following elements:

- \* Source media stream, in the form of consecutive raw video frames and audio samples;
- \* Media encoder with rate control capabilities. It takes the source media stream and encodes it to an RTP stream at a target bit rate  $R_v$ . Note that the actual output rate from the encoder  $R_o$  may fluctuate around the target  $R_v$ . Also, the encoder can only change its rate at rather coarse time intervals, e.g., once every 0.5 seconds.
- \* RTP sender, responsible for calculating the target bit rate  $R_n$  based on network congestion indicators (delay, loss, or ECN marking reports from the receiver), for updating the video encoder with a new target rate  $R_v$ , and for regulating the

actual sending rate  $R_s$  accordingly. A rate shaping buffer is employed to absorb the instantaneous difference between video encoder output rate  $R_v$  and sending rate  $R_s$ . The buffer size  $L_s$ , together with  $R_n$ , influences the calculation of actual sending rate  $R_s$  and video encoder target rate  $R_v$ . The RTP sender also generates RTP timestamp in outgoing packets.

\* RTP receiver, responsible for measuring and estimating end-to-end delay based on sender RTP timestamp. In the presence of packet loss and ECN markings, it keeps track of packet loss and ECN marking ratios. It calculates the equivalent delay  $x_n$  that accounts for queuing delay, ECN marking, and packet loss, as well as the derivative (i.e., rate of change) of this congestion signal as  $x'_n$ . The receiver feeds both pieces of information ( $x_n$  and  $x'_n$ ) back to the sender via periodic RTCP reports.

\* Network node, with several modes of operation. The system can work with the default behavior of a simple drop tail queue. It can also benefit from advanced AQM features such as RED-based ECN marking, and PCN marking using a token bucket algorithm. Note that network node operation is out of scope for the design of NADA.

In the following, we will elaborate on the respective operations at the NADA receiver and sender.

#### 4. NADA Receiver Behavior

The receiver continuously monitors end-to-end per-packet statistics in terms of delay, loss, and/or ECN marking ratios. It then aggregates all forms of congestion indicators into the form of an equivalent delay and periodically reports this back to the sender. In addition, the receiver tracks the receiving rate of the flow and includes that in the feedback message.

##### 4.1 Estimation of one-way delay and queuing delay

The delay estimation process in NADA follows a similar approach as in earlier delay-based congestion control schemes, such as LEDBAT [RFC6817]. NADA estimates the forward delay as having a constant base delay component plus a time varying queuing delay component. The base delay is estimated as the minimum value of one-way delay observed over a relatively long period (e.g., tens of minutes), whereas the individual queuing delay value is taken to be the difference between one-way delay and base delay.

In mathematical terms, for packet  $n$  arriving at the receiver, one-way delay is calculated as:

$$x_n = t_{r,n} - t_{s,n},$$

where  $t_{s,n}$  and  $t_{r,n}$  are sender and receiver timestamps, respectively. A real-world implementation should also properly handle practical issues such as wrap-around in the value of  $x_n$ , which are omitted from the above simple expression for brevity.

The base delay,  $d_f$ , is estimated as the minimum value of previously observed  $x_n$ 's over a relatively long period. This assumes that the drift between sending and receiving clocks remains bounded by a small value.

Correspondingly, the queuing delay experienced by the packet  $n$  is estimated as:

$$d_n = x_n - d_f.$$

The individual sample values of queuing delay should be further filtered against various non-congestion-induced noise, such as spikes due to processing "hiccup" at the network nodes. We denote the resulting queuing delay value as  $d_{\hat{n}}$ .

Our current implementation employs a simple 5-point median filter over per-packet queuing delay estimates, followed by an exponential smoothing filter. We have found such relatively simple treatment to suffice in guarding against processing delay outliers observed in wired connections. For wireless connections with a higher packet delay variation (PDV), more sophisticated techniques on de-noising, outlier rejection, and trend analysis may be needed.

Like other delay-based congestion control schemes, performance of NADA depends on the accuracy of its delay measurement and estimation module. Appendix A in [RFC6817] provides an extensive discussion on this aspect.

#### 4.2 Estimation of packet loss/marketing ratio

The receiver detects packet losses via gaps in the RTP sequence numbers of received packets. It then calculates instantaneous packet loss ratio as the ratio between the number of missing packets over the number of total transmitted packets in the given time window (e.g., during the most recent 500ms). This instantaneous value is passed over an exponential smoothing filter, and the filtered result is reported back to the sender as the observed packet loss ratio  $p_L$ .

We note that more sophisticated methods in packet loss ratio calculation, such as that adopted by TFRC [Floyd-CCR00], will likely be beneficial. These alternatives are currently under investigation.

Estimation of packet marking ratio  $p_M$ , when ECN is enabled at bottleneck network nodes along the path, will follow the same procedure as above. Here it is assumed that ECN marking information at the IP header are somehow passed along to the transport layer by the receiving endpoint.

#### 4.3 Non-linear warping of delay

In order for a delay-based flow to hold its ground and sustain a reasonable share of bandwidth in the presence of a loss-based flow (e.g., loss-based TCP), it is important to distinguish between different levels of observed queuing delay. For instance, a moderate queuing delay value below 100ms is likely self-inflicted or induced by other delay-based flows, whereas a high queuing delay value of several hundreds of milliseconds may indicate the presence of a loss-based flow that does not refrain from increased delay.

Inspired by the delay-adaptive congestion window backoff policy in [Budzisz-TON11] -- the work by itself is a window-based congestion control scheme with fair coexistence with TCP -- we devise the following non-linear warping of estimated queuing delay value:

$$\begin{aligned} d_{\text{tilde}_n} &= (d_{\text{hat}_n}), \quad \text{if } d_{\text{hat}_n} < d_{\text{th}}; \\ d_{\text{tilde}_n} &= d_{\text{th}} \frac{(d_{\text{max}} - d_{\text{hat}_n})^4}{(d_{\text{max}} - d_{\text{th}})^4}, \quad \text{if } d_{\text{th}} < d_{\text{hat}_n} < d_{\text{max}}; \\ d_{\text{tilde}_n} &= 0, \quad \text{if } d_{\text{hat}_n} > d_{\text{max}}. \end{aligned}$$

Here, the queuing delay value is unchanged when it is below the first threshold  $d_{\text{th}}$ ; it is discounted following a non-linear curve when its value falls between  $d_{\text{th}}$  and  $d_{\text{max}}$ ; above  $d_{\text{max}}$ , the high queuing delay value no longer counts toward congestion control.

When queuing delay is in the range  $(0, d_{\text{th}})$ , NADA operates in pure delay-based mode if no losses/markings are present. When queuing delay exceeds  $d_{\text{max}}$ , NADA reacts to loss/markings only. In between  $d_{\text{th}}$  and  $d_{\text{max}}$ , the sending rate will converge and stabilize at an operating point with a fairly high queuing delay and non-zero packet loss ratio.

In our current implementation  $d_{\text{th}}$  is chosen as 50ms and  $d_{\text{max}}$  is chosen as 400ms. The impact of the choice of  $d_{\text{th}}$  and  $d_{\text{max}}$  will be investigated in future work.

#### 4.4 Aggregating congestion signals

The receiver aggregates all three forms of congestion signal in terms of an equivalent delay:

$$x_n = d_{\text{tilde}_n} + p_M d_M + p_L d_L, \quad (1)$$

where  $d_M$  is a prescribed fictitious delay value associated with ECN markings (e.g.,  $d_M = 200$  ms), and  $d_L$  is a prescribed fictitious delay value associated with packet losses (e.g.,  $d_L = 1$  second). By introducing a large fictitious delay penalty for ECN marking and packet loss, the proposed scheme leads to low end-to-end actual delay in the presence of such events.

While the value of  $d_M$  and  $d_L$  are fixed and predetermined in the current design, a scheme for automatically tuning these values based on desired bandwidth sharing behavior in the presence of other competing loss-based flows (e.g., loss-based TCP) is being studied.

In the absence of ECN marking from the network, the value of  $x_n$  falls back to the observed queuing delay  $d_n$  for packet  $n$  when queuing delay is low and no packets are lost over a lightly congested path. In that case the algorithm operates in purely delay-based mode.

#### 4.5 Estimating receiving rate

Estimation of receiving rate of the flow is fairly straightforward. NADA maintains a recent observation window of 500ms, and simply divides the total size of packets arriving during that window over the time span. The receiving rate is denoted as  $R_r$ .

#### 4.6 Sending periodic feedback

Periodically, the receiver feeds back a tuple of the most recent values of  $\langle d_{\text{hat}_n}, x_n, x'_n, R_r \rangle$  in RTCP feedback messages to aid the sender in its calculation of target rate. The queuing delay value  $d_{\text{hat}_n}$  is included along with the composite congestion signal  $x_n$  so that the sender can decide whether the network is truly underutilized (see Sec. 6.1.1 Accelerated ramp-up).

The value of  $x'_n$  corresponds to the derivative (i.e., rate of change) of the composite congestion signal:

$$x'_n = \frac{x_n - x_{(n-k)}}{\text{delta}}, \quad (2)$$

where the interval between consecutive RTCP feedback messages is denoted as  $\delta$ . The packet indices corresponding to the current and previous feedback are  $n$  and  $(n-k)$ , respectively.

The choice of target feedback interval needs to strike the right balance between timely feedback and low RTCP feedback message counts. Through simulation studies and frequency-domain analysis, it was determined that a feedback interval below 250ms will not break up the feedback control loop of the NADA congestion control algorithm. Thus, it is recommended to use a target feedback interval of 100ms. This will result in a feedback bandwidth of 16Kbps with 200 bytes per feedback message, less than 0.1% overhead for a 1Mbps flow.

#### 4.7 Discussions on delay metrics

The current design works with relative one-way-delay (OWD) as the main indication of congestion. The value of the relative OWD is obtained by maintaining the minimum value of observed OWD over a relatively long time horizon and subtract that out from the observed absolute OWD value. Such an approach cancels out the fixed difference between the sender and receiver clocks. It has been widely adopted by other delay-based congestion control approaches such as LEDBAT [RFC6817]. As discussed in [RFC6817], the time horizon for tracking the minimum OWD needs to be chosen with care: it must be long enough for an opportunity to observe the minimum OWD with zero queuing delay along the path, and sufficiently short so as to timely reflect "true" changes in minimum OWD introduced by route changes and other rare events.

The potential drawback in relying on relative OWD as the congestion signal is that when multiple flows share the same bottleneck, the flow arriving late at the network experiencing a non-empty queue may mistakenly consider the standing queuing delay as part of the fixed path propagation delay. This will lead to slightly unfair bandwidth sharing among the flows.

Alternatively, one could move the per-packet statistical handling to the sender instead and use RTT in lieu of OWD, assuming that per-packet ACKs are available. The main drawback of this latter approach is that the scheme will be confused by congestion in the reverse direction.

Note that the choice of either delay metric (relative OWD vs. RTT) involves no change in the proposed rate adaptation algorithm at the sender. Therefore, comparing the pros and cons regarding which delay metric to adopt can be kept as an orthogonal direction of investigation.

## 5. NADA Sender Behavior

Figure 1 provides a detailed view of the NADA sender. Upon receipt of an RTCP report from the receiver, the NADA sender updates its calculation of the reference rate  $R_n$ . It further adjusts both the target rate for the live video encoder  $R_v$  and the sending rate  $R_s$  over the network based on the updated value of  $R_n$ , as well as the size of the rate shaping buffer.

In the following, we describe these modules in further detail, and explain how they interact with each other.

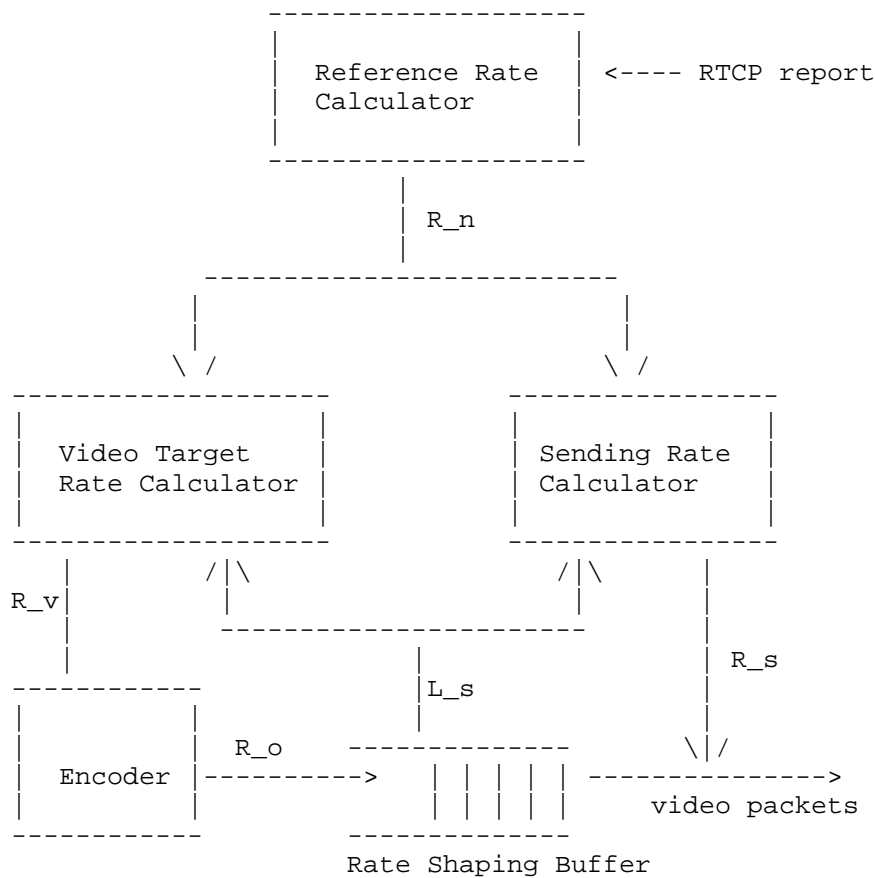


Figure 1 NADA Sender Structure

### 5.1 Reference rate calculation

The sender initializes the reference rate  $R_n$  as  $R_{\min}$  by default, or to a value specified by the upper-layer application. [Editor's note: should proper choice of starting rate value be within the scope of the CC solution? ]

The reference rate  $R_n$  is calculated based on receiver feedback information regarding queuing delay  $d_{\text{tilde}_n}$ , composite congestion signal  $x_n$ , its derivative  $x'_n$ , as well as the receiving rate  $R_r$ . The sender switches between two modes of operation:

- \* Accelerated ramp up: if the reported queuing delay is close to zero and both values of  $x_n$  and  $x'_n$  are close to zero, indicating empty queues along the path of the flow and, consequently, underutilized network bandwidth; or

- \* Gradual rate update: in all other conditions, whereby the receiver reports on a standing or increasing/decreasing queue and/or composite congestion signal.

#### 5.1.1 Accelerated ramp up

In the absence of a non-zero congestion signal to guide the sending rate calculation, the sender needs to ramp up its estimated bandwidth as quickly as possible without introducing excessive queuing delay. Ideally the flow should inflict no more than  $T_{\text{th}}$  milliseconds of queuing delay at the bottleneck during the ramp-up process. A typical value of  $T_{\text{th}}$  is 50ms.

Note that the sender will be aware of any queuing delay introduced by its rate increase after at least one round-trip time. In addition, the bottleneck bandwidth  $C$  is greater than or equal to the receive rate  $R_r$  reported from the most recent "no congestion" feedback message. The rate  $R_n$  is updated as follows:

$$\gamma = \min \left[ \gamma_0, \frac{T_{\text{th}}}{\text{RTT}_0 + \delta_0} \right] \quad (3)$$

$$R_n = (1 + \gamma) R_r \quad (4)$$

In (3) and (4), the multiplier  $\gamma$  for rate increase is upper-bounded by a fixed ratio  $\gamma_0$  (e.g., 20%), as well as a ratio which depends

on  $T_{th}$ , base RTT as measured during the non-congested phase, and target ACK interval  $\delta_0$ . The rationale behind this is that the rate increase multiplier should decrease with the delay in the feedback control loop, and that  $RTT_0 + \delta_0$  provides a worst-case estimate of feedback control delay when the network is not congested.

#### 5.1.2. Gradual rate update

When the receiver reports indicate a standing congestion level, NADA operates in gradual update mode, and calculates its reference rate as:

$$R_n \leftarrow R_n + \frac{\kappa * \delta_s}{\tau_o^2} * (\theta - (R_n - R_{min}) * x_{hat}) \quad (5)$$

where

$$\theta = w * (R_{max} - R_{min}) * x_{ref}. \quad (6)$$

$$x_{hat} = x_n + \eta * \tau_o * x'_n \quad (7)$$

In (5),  $\delta_s$  refers to the time interval between current and previous rate updates. Note that  $\delta_s$  is the same as the RTCP report interval at the receiver (see  $\delta$  from (2)) when the backward path is uncongested.

In (6),  $R_{min}$  and  $R_{max}$  denote the content-dependent rate range the encoder can produce. The weighting factor reflecting a flow's priority is  $w$ . The reference congestion signal  $x_{ref}$  is chosen so that the maximum rate of  $R_{max}$  can be achieved when  $x_{hat} = w * x_{ref}$ .

Proper choice of the scaling parameters  $\eta$  and  $\kappa$  in (5) and (7) can ensure system stability so long as the RTT falls below the upper bound of  $\tau_o$ . The recommended default value of  $\tau_o$  is chosen as 500ms.

For both modes of operations, the final reference rate  $R_n$  is clipped within the range of  $[R_{min}, R_{max}]$ . Note also that the sender does not need any explicit knowledge of the management scheme inside the network. Rather, it reacts to the aggregation of all forms of congestion indications (delay, loss, and explicit markings) via the composite congestion signals  $x_n$  and  $x'_n$  from the receiver in a coherent manner.

## 5.2 Video encoder rate control

The video encoder rate control procedure has the following characteristics:

- \* Rate changes can happen only at large intervals, on the order of seconds.
- \* The encoder output rate may fluctuate around the target rate  $R_v$ .
- \* The encoder output rate is further constrained by video content complexity. The range of the final rate output is  $[R_{min}, R_{max}]$ . Note that it is content-dependent and may vary over time.

The operation of the live video encoder is out of the scope of the design for the congestion control scheme in NADA. Instead, its behavior is treated as a black box.

## 5.3 Rate shaping buffer

A rate shaping buffer is employed to absorb any instantaneous mismatch between encoder rate output  $R_o$  and regulated sending rate  $R_s$ . The size of the buffer evolves from time  $t-\tau$  to time  $t$  as:

$$L_s(t) = \max [0, L_s(t-\tau) + (R_o - R_s) \cdot \tau].$$

A large rate shaping buffer contributes to higher end-to-end delay, which may harm the performance of real-time media communications. Therefore, the sender has a strong incentive to constrain the size of the shaping buffer. It can either deplete it faster by increasing the sending rate  $R_s$ , or limit its growth by reducing the target rate for the video encoder rate control  $R_v$ .

## 5.4 Adjusting video target rate and sending rate

The target rate for the live video encoder is updated based on both the reference rate  $R_n$  and the rate shaping buffer size  $L_s$ , as follows:

$$R_v = R_n - \beta_v \cdot \frac{L_s}{\tau_v}. \quad (8)$$

Similarly, the outgoing rate is regulated based on both the reference rate  $R_n$  and the rate shaping buffer size  $L_s$ , such that:

$$R_s = R_n + \beta_s \cdot \frac{L_s}{\tau_v}. \quad (9)$$

In (8) and (9), the first term indicates the rate calculated from network congestion feedback alone. The second term indicates the influence of the rate shaping buffer. A large rate shaping buffer nudges the encoder target rate slightly below -- and the sending rate slightly above -- the reference rate  $R_n$ .

Intuitively, the amount of extra rate offset needed to completely drain the rate shaping buffer within the same time frame of encoder rate adaptation  $\tau_v$  is given by  $L_s/\tau_v$ . The scaling parameters  $\beta_v$  and  $\beta_s$  can be tuned to balance between the competing goals of maintaining a small rate shaping buffer and deviating the system from the reference rate point.

## 6. Incremental Deployment

One nice property of NADA is the consistent video endpoint behavior irrespective of network node variations. This facilitates gradual, incremental adoption of the scheme.

To start off with, the encoder congestion control mechanism can be implemented without any explicit support from the network, and relies solely on observed one-way delay measurements and packet loss ratios as implicit congestion signals.

When ECN is enabled at the network nodes with RED-based marking, the receiver can fold its observations of ECN markings into the calculation of the equivalent delay. The sender can react to these explicit congestion signals without any modification.

Ultimately, networks equipped with proactive marking based on token bucket level metering can reap the additional benefits of zero standing queues and lower end-to-end delay and work seamlessly with existing senders and receivers.

## 7. Implementation Status

The NADA scheme has been implemented in the ns-2 simulation platform [ns2]. Extensive simulation evaluations of an earlier version of the draft are documented in [Zhu-PV13]. Evaluation results of the current draft over several test cases in [I-D.draft-sarker-rmcat-eval-test] have been presented at recent IETF meetings [IETF-90][IETF-91].

The scheme has also been implemented and evaluated in a lab setting as described in [IETF-90]. Preliminary evaluation results of NADA in single-flow and multi-flow scenarios have been presented in [IETF-91].

## 8. IANA Considerations

There are no actions for IANA.

## 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.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

### 9.2 Informative References

- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.
- [RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and Kuehlewind, M., "Low Extra Delay Background Transport (LED BAT)", RFC 6817, December 2012.
- [Floyd-CCR00] Floyd, S., Handley, M., Padhye, J., and Widmer, J., "Equation-based Congestion Control for Unicast Applications", ACM SIGCOMM Computer Communications Review, vol. 30. no. 4., pp. 43-56, October 2000.
- [Budzisz-TON11] Budzisz, L. et al., "On the Fair Coexistence of Loss- and Delay-Based TCP", IEEE/ACM Transactions on Networking, vol. 19, no. 6, pp. 1811-1824, December 2011.

- [ns2] "The Network Simulator - ns-2", <http://www.isi.edu/nsnam/ns/>
- [Zhu-PV13] Zhu, X. and Pan, R., "NADA: A Unified Congestion Control Scheme for Low-Latency Interactive Video", in Proc. IEEE International Packet Video Workshop (PV'13). San Jose, CA, USA. December 2013.
- [I-D.draft-sarker-rmcat-eval-test] Sarker, Z., Singh, V., Zhu, X., and Ramalho, M., "Test Cases for Evaluating RMCAT Proposals", draft-sarker-rmcat-eval-test-01 (work in progress), June 2014.
- [IETF-90] Zhu, X. et al., "NADA Update: Algorithm, Implementation, and Test Case Evaluation Results", presented at IETF 90, <https://tools.ietf.org/agenda/90/slides/slides-90-rmcat-6.pdf>
- [IETF-91] Zhu, X. et al., "NADA Algorithm Update and Test Case Evaluations", presented at IETF 91 Interim, <https://datatracker.ietf.org/meeting/91/agenda/rmcat/>

#### Appendix A. Network Node Operations

NADA can work with different network queue management schemes and does not assume any specific network node operation. As an example, this appendix describes three variations of queue management behavior at the network node, leading to either implicit or explicit congestion signals.

In all three flavors described below, the network queue operates with the simple first-in-first-out (FIFO) principle. There is no need to maintain per-flow state. Such a simple design ensures that the system can scale easily with a large number of video flows and high link capacity.

NADA sender behavior stays the same in the presence of all types of congestion indicators: delay, loss, ECN marking due to either RED/ECN or PCN algorithms. This unified approach allows a graceful transition of the scheme as the network shifts dynamically between light and heavy congestion levels.

### A.1 Default behavior of drop tail

In a conventional network with drop tail or RED queues, congestion is inferred from the estimation of end-to-end delay and/or packet loss. Packet drops at the queue are detected at the receiver, and contributes to the calculation of the equivalent delay  $x_n$ . No special action is required at network node.

### A.2 ECN marking

In this mode, the network node randomly marks the ECN field in the IP packet header following the Random Early Detection (RED) algorithm [RFC2309]. Calculation of the marking probability involves the following steps:

\* upon packet arrival, update smoothed queue size  $q_{avg}$  as:

$$q_{avg} = \alpha * q + (1 - \alpha) * q_{avg}.$$

The smoothing parameter  $\alpha$  is a value between 0 and 1. A value of  $\alpha=1$  corresponds to performing no smoothing at all.

\* calculate marking probability  $p$  as:

$$p = 0, \text{ if } q < q_{lo};$$

$$p = p_{max} * \frac{q_{avg} - q_{lo}}{q_{hi} - q_{lo}}, \text{ if } q_{lo} \leq q < q_{hi};$$

$$p = 1, \text{ if } q \geq q_{hi}.$$

Here,  $q_{lo}$  and  $q_{hi}$  corresponds to the low and high thresholds of queue occupancy. The maximum marking probability is  $p_{max}$ .

The ECN markings events will contribute to the calculation of an equivalent delay  $x_n$  at the receiver. No changes are required at the sender.

### A.3 PCN marking

As a more advanced feature, we also envisage network nodes which support PCN marking based on virtual queues. In such a case, the marking probability of the ECN bit in the IP packet header is calculated as follows:

```
* upon packet arrival, meter packet against token bucket (r,b);  
* update token level b_tk;  
* calculate the marking probability as:  
  
  p = 0, if b-b_tk < b_lo;  
  
  p = p_max *  $\frac{b-b_{tk}-b_{lo}}{b_{hi}-b_{lo}}$ , if b_lo <= b-b_tk < b_hi;  
  
  p = 1, if b-b_tk >= b_hi.
```

Here, the token bucket lower and upper limits are denoted by  $b_{lo}$  and  $b_{hi}$ , respectively. The parameter  $b$  indicates the size of the token bucket. The parameter  $r$  is chosen as  $r=\gamma \cdot C$ , where  $\gamma < 1$  is the target utilization ratio and  $C$  designates link capacity. The maximum marking probability is  $p_{max}$ .

The ECN markings events will contribute to the calculation of an equivalent delay  $x_n$  at the receiver. No changes are required at the sender. The virtual queuing mechanism from the PCN marking algorithm will lead to additional benefits such as zero standing queues.

#### Authors' Addresses

Xiaoqing Zhu  
Cisco Systems,  
12515 Research Blvd.,  
Austin, TX 78759, USA  
Email: xiaoqzhu@cisco.com

Rong Pan  
Cisco Systems  
510 McCarthy Blvd,  
Milpitas, CA 95134, USA  
Email: ropan@cisco.com

Michael A. Ramalho  
6310 Watercrest Way Unit 203  
Lakewood Ranch, FL, 34202, USA  
Email: mramalho@cisco.com

Sergio Mena de la Cruz  
Cisco Systems  
EPFL, Quartier de l'Innovation, Batiment E  
Ecublens, Vaud 1015, Switzerland  
Email: [semena@cisco.com](mailto:semena@cisco.com)

Charles Ganzhorn  
7900 International Drive  
International Plaza, Suite 400  
Bloomington, MN 55425, USA  
Email: [charles.ganzhorn@gmail.com](mailto:charles.ganzhorn@gmail.com)

Paul E. Jones  
7025 Kit Creek Rd.  
Research Triangle Park, NC 27709, USA  
Email: [paulej@packetizer.com](mailto:paulej@packetizer.com)

Stefano D'Aronco  
EPFL STI IEL LTS4  
ELD 220 (Batiment ELD), Station 11  
CH-1015 Lausanne, Switzerland  
Email: [stefano.daronco@epfl.ch](mailto:stefano.daronco@epfl.ch)