Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 31, 2015

S. Holmer
H. Lundin
Google
G. Carlucci
L. De Cicco
S. Mascolo
Politecnico di Bari

June 29, 2015

A Google Congestion Control Algorithm for Real-Time Communication draft-alvestrand-rmcat-congestion-03

Abstract

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

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

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 31, 2015.

Table of Contents

1. Introduction

Congestion control is a requirement for all applications sharing the Internet resources [RFC2914].

Congestion control for real-time media is challenging for a number of reasons:

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

This memo describes two congestion control algorithms that together are able to provide good performance and reasonable bandwidth sharing with other video flows using the same congestion control and with TCP flows that share the same links.

The signaling used consists of experimental RTP header extensions and RTCP messages RFC 3550 [RFC3550] as defined in [abs-send-time], [I-D.alvestrand-rmcat-remb] and [I-D.holmer-rmcat-transport-wide-cc-extensions].

1.1. Mathematical notation conventions

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

The following notational conventions are used:

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

X_hat An estimate of the true value of variable X - conventionally marked by a circumflex accent on top of the variable name.

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

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

X_bar^T The transpose of vector X_bar.

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

2. System model

The following elements are in the system:

- RTP packet - an RTP packet containing media data.
- Packet group - a set of RTP packets transmitted from the sender uniquely identified by the group departure and group arrival time (absolute send time) [abs-send-time]. These could be video packets, audio packets, or a mix of audio and video packets.
- Incoming media stream - a stream of frames consisting of RTP packets.
- RTP sender - sends the RTP stream over the network to the RTP receiver. It generates the RTP timestamp and the abs-send-time header extension
- RTP receiver - receives the RTP stream, marks the time of arrival.
- RTCP sender at RTP receiver - sends receiver reports, REMB messages and transport-wide RTCP feedback messages.
- RTCP receiver at RTP sender - receives receiver reports and REMB messages and transport-wide RTCP feedback messages, reports these to the sender side controller.
- RTCP receiver at RTP receiver, receives sender reports from the sender.
- Loss-based controller - takes loss rate measurement, round trip time measurement and REMB messages, and computes a target sending bitrate.
- Delay-based controller - takes the packet arrival info, either at the RTP receiver, or from the feedback received by the RTP sender, and computes a maximum bitrate which it passes to the loss-based controller.

Together, loss-based controller and delay-based controller implement the congestion control algorithm.

3. Feedback and extensions

There are two ways to implement the proposed algorithm. One where both the controllers are running at the send-side, and one where the delay-based controller runs on the receive-side and the loss-based controller runs on the send-side.

The first version can be realized by using a per-packet feedback protocol as described in [I-D.holmer-rmcat-transport-wide-cc-extensions]. Here, the RTP receiver will record the arrival time and the transport-wide sequence number of each received packet, which will be sent back to the sender periodically using the transport-wide feedback message. The RECOMMENDED feedback interval is once per received video frame or at least once every 30 ms if audio-only or multi-stream. If the feedback overhead needs to be limited this interval can be increased to 100 ms.

The sender will map the received {sequence number, arrival time} pairs to the send-time of each packet covered by the feedback report, and feed those timestamps to the delay-based controller. It will also compute a loss ratio based on the sequence numbers in the feedback message.

The second version can be realized by having a delay-based controller at the receive-side, monitoring and processing the arrival time and size of incoming packets. The sender SHOULD use the abs-send-time RTP header extension [abs-send-time] to enable the receiver to compute the inter-group delay variation. The output from the delay-based controller will be a bitrate, which will be sent back to the sender using the REMB feedback message [I-D.alvestrand-rmcat-remb]. The packet loss ratio is sent back via RTCP receiver reports. At the sender the bitrate in the REMB message and the fraction of packets lost are fed into the loss-based controller, which outputs a final target bitrate. It is RECOMMENDED to send the REMB message as soon as congestion is detected, and otherwise at least once every second.

4. Delay-based control

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

4.1. Arrival-time model

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

We define the inter-arrival time, t(i) - t(i-1), as the difference in arrival time of two packets or two groups of packets. Correspondingly, the inter-departure time, T(i) - T(i-1), is defined as the difference in departure-time of two packets or two groups of packets. Finally, the inter-group delay variation, d(i), is defined as the difference between the inter-arrival time and the inter-departure time. Or interpreted differently, as the difference between the delay of group i and group i-1.

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

At the receiving side we are observing groups of incoming packets, where a group of packets is defined as follows:

- A sequence of packets which are sent within a burst_time interval constitute a group. RECOMMENDED value for burst_time is 5 ms.

- In addition, any packet which has an inter-arrival time less than burst_time and an inter-group delay variation d(i) less than 0 is also considered being part of the current group of packets. The reasoning behind including these packets in the group is to better handle delay transients, caused by packets being queued up for reasons unrelated to congestion. As an example this has been observed to happen on many Wi-Fi and wireless networks.

An inter-departure time is computed between consecutive groups as T(i) - T(i-1), where T(i) is the departure timestamp of the last packet in the current packet group being processed. Any packets received out of order are ignored by the arrival-time model.

Each group is assigned a receive time t(i), which corresponds to the time at which the last packet of the group was received. A group is delayed relative to its predecessor if t(i) - t(i-1) > T(i) - T(i-1), i.e., if the inter-arrival time is larger than the inter-departure time.

Since the time ts to send a group of packets of size L over a path with a capacity of C is roughly

$$ts = L/C$$

we can model the inter-group delay variation as:

$$d(i) = L(i)/C(i) - L(i-1)/C(i-1) + w(i) =$$

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

Here, w(i) is a sample from a stochastic process W, which is a function of the capacity C(i), the current cross traffic, and the current sent bitrate. C is modeled as being constant as we expect it to vary more slowly than other parameters of this model. We model W as a white Gaussian process. If we are over-using the channel we expect the mean of w(i) to increase, and if a queue on the network path is being emptied, the mean of 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 1

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

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

4.2. Arrival-time filter

The parameters d(i) and dL(i) are readily available for each group of packets, i > 1, and we want to estimate C(i) and m(i) and use those estimates to detect whether or not the bottleneck link is over-used. These parameters can be estimated by any adaptive filter - we are using the Kalman filter.

Let

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

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

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

where $\bar{u}(i)$ is the state noise that we model as a stationary process with Gaussian statistic with zero mean and covariance

$$Q(i) = E\{u\_bar(i) * u\_bar(i)^T\}$$

Q(i) is RECOMMENDED as a diagonal matrix with main diagonal elements as:

$$diag(Q(i)) = [10^{-13}\ 10^{-3}]^T$$

Given equation 1 we get

$$d(i) = h\_bar(i)^T * theta\_bar(i) + v(i)$$

$$h\_bar(i) = [dL(i)\ \ 1]^T$$

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

The Kalman filter recursively updates our estimate

$$theta\_hat(i) = [1/C\_hat(i)\ m\_hat(i)]^T$$

as

$$z(i) = d(i) - h\_bar(i)^T * theta\_hat(i-1)$$

$$theta\_hat(i) = theta\_hat(i-1) + z(i) * k\_bar(i)$$

$$k\_bar(i) = \frac{(E(i-1) + Q(i)) * h\_bar(i)}{var\_v\_hat(i) + h\_bar(i)^T * (E(i-1) + Q(i)) * h\_bar(i)}$$

$$E(i) = (I - k\_bar(i) * h\_bar(i)^T) * (E(i-1) + Q(i))$$

where I is the 2-by-2 identity matrix.

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

$$var\_v\_hat(i) = max(beta * var\_v\_hat(i-1) + (1-beta) * z(i)^2,\ 1)$$

$$beta = (1-chi)^{(30/(1000 * f\_max))}$$

where f_max = max {1/(T(j) - T(j-1))} for j in i-K+1,...,i is the highest rate at which the last K packet groups have been received and chi is a filter coefficient typically chosen as a number in the interval [0.1, 0.001]. Since our assumption that v(i) should be zero mean WGN is less accurate in some cases, we have introduced an additional outlier filter around the updates of var_v_hat. If z(i) > 3*sqrt(var_v_hat) the filter is updated with 3*sqrt(var_v_hat) rather

than z(i). For instance v(i) will not be white in situations where packets are sent at a higher rate than the channel capacity, in which case they will be queued behind each other.

4.3. Over-use detector

The offset estimate m(i), obtained as the output of the arrival-time filter, 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. A definitive over-use will be signaled only if over-use has been detected for at least gamma_2 milliseconds. However, if m(i) < m(i-1), over-use will not be signaled even if all the above conditions are met. Similarly, the opposite state, under-use, is detected when m(i) < -gamma_1(i). If neither over-use nor under-use is detected, the detector will be in the normal state.

The threshold gamma_1 has a remarkable impact on the overall dynamics and performance of the algorithm. In particular, it has been shown that using a static threshold 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 gamma_1 to a sufficiently large value.

The reason is that, by using a larger value of gamma_1, a larger queuing delay can be tolerated, whereas with a small 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 the delay-based estimate of the available bandwidth A_hat (see Section 4.4). Thus, it is necessary to dynamically tune the threshold 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 gamma_1(i) according to the following dynamic equation:

$$\gamma_1(i) = \gamma_1(i-1) + (t(i)-t(i-1)) \cdot K(i) \cdot (|m(i)|-\gamma_1(i-1))$$

with K(i)=K_d if |m(i)| < gamma_1(i-1) or K(i)=K_u otherwise. The rationale is to increase gamma_1(i) when m(i) is outside of the range [-gamma_1(i-1),gamma_1(i-1)], whereas, when the offset estimate m(i) falls back into the range, gamma_1 is decreased. In this way when m(i) increases, for instance due to a TCP flow entering the same bottleneck, 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 [Pv13]. Moreover, gamma_1(i) SHOULD NOT be updated if this condition holds:

$$|m(i)| - \gamma_1(i) > 15$$

It is also RECOMMENDED to clamp $\gamma_1(i)$ to the range [6, 600], since a too small $\gamma_1(i)$ can cause the detector to become overly sensitive.

On the other hand, when $m(i)$ falls back into the range $[-\gamma_1(i-1), \gamma_1(i-1)]$ the threshold $\gamma_1(i)$ is decreased so that a lower queuing delay can be achieved.

It is RECOMMENDED to choose $K_u > K_d$ so that the rate at which γ_1 is increased is higher than the rate at which it is decreased. With this setting it is possible to increase the threshold in the case of a concurrent TCP flow and prevent starvation as well as enforcing intra-protocol fairness. RECOMMENDED values for $\gamma_1(0)$, γ_2, K_u and K_d are respectively 12.5 ms, 10 ms, 0.01 and 0.00018.

4.4. Rate control

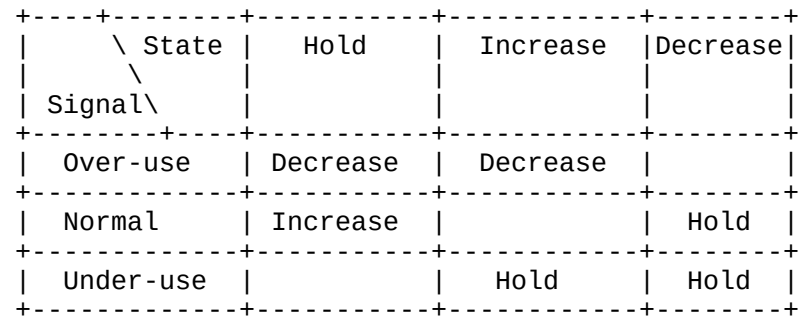
The rate control is split in two parts, one controlling the bandwidth estimate based on delay, and one controlling the bandwidth estimate based on loss. Both are designed to increase the estimate of the available bandwidth A_{hat} as long as there is no detected congestion and to ensure that we will eventually match the available bandwidth of the channel and detect an over-use.

As soon as over-use has been detected, the available bandwidth estimated by the delay-based controller 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:



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 delay-based estimate of the available bandwidth is increased, either multiplicatively or additively, depending on its current state.

The system does a multiplicative increase if the current bandwidth estimate appears to be far from convergence, while it does an additive increase if it appears to be closer to convergence. We assume that we are close to convergence if the currently incoming bitrate, R_hat(i), is close to an average of the incoming bitrates at the time when we previously have been in the Decrease state. "Close" is defined as three standard deviations around this average. It is RECOMMENDED to measure this average and standard deviation with an exponential moving average with the smoothing factor 0.95, as it is expected that this average covers multiple occasions at which we are in the Decrease state. Whenever valid estimates of these statistics are not available, we assume that we have not yet come close to convergence and therefore remain in the multiplicative increase state.

If R_hat(i) increases above three standard deviations of the average max bitrate, we assume that the current congestion level has changed, at which point we reset the average max bitrate and go back to the multiplicative increase state.

R_hat(i) is the incoming bitrate measured by the delay-based controller over a T seconds window:

$$\hat{R}(i) = \frac{1}{T} \cdot \sum_{j=1}^{N(i)} L(j)$$

N(i) is the number of packets received the past T seconds and L(j) is the payload size of packet j. A window between 0.5 and 1 second is RECOMMENDED.

During multiplicative increase, the estimate is increased by at most 8% per second.

```
eta = 1.08^min(time_since_last_update_ms / 1000, 1.0)
A_hat(i) = eta * A_hat(i-1)
```

During the additive increase the estimate is increased with at most half a packet per response_time interval. The response_time interval is estimated as the round-trip time plus 100 ms as an estimate of over-use estimator and detector reaction time.

```
response_time_ms = 100 + rtt_ms
beta = 0.5 * min(time_since_last_update_ms / response_time_ms, 1.0)
A_hat(i) = A_hat(i-1) + max(1000, beta * expected_packet_size_bits)
```

expected_packet_size_bits is used to get a slightly slower slope for the additive increase at lower bitrates. It can for instance be computed from the current bitrate by assuming a frame rate of 30 frames per second:

```
bits_per_frame = A_hat(i-1) / 30
packets_per_frame = ceil(bits_per_frame / (1200 * 8))
avg_packet_size_bits = bits_per_frame / packets_per_frame
```

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

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

When an over-use is detected the system transitions to the decrease state, where the delay-based available bandwidth estimate is decreased to a factor times the currently incoming bitrate.

```
A_hat(i) = alpha * R_hat(i)
```

alpha is typically chosen to be in the interval [0.8, 0.95], 0.85 is the RECOMMENDED value.

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 A_hat 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.

It is RECOMMENDED that the routine to update A_hat(i) is run at least once every response_time interval.

4.5. Parameters settings

Parameter	Description	RECOMMENDED Value
burst_time	Time limit in milliseconds between packet bursts which identifies a group	5 ms
Q	State noise covariance matrix	diag(Q(i)) = [10^-13 10^-3]^T
E(0)	Initial value of the system error covariance	diag(E(0)) = [100 0.1]^T
chi	Coefficient used for the measured noise variance	[0.1, 0.001]
gamma_1(0)	Initial value for the adaptive threshold	12.5 ms
gamma_2	Time required to trigger an overuse signal	10 ms
K_u	Coefficient for the adaptive threshold	0.01
K_d	Coefficient for the adaptive threshold	0.00018
T	Time window for measuring the received bitrate	[0.5, 1] s
alpha	Decrease rate factor	0.85

Table 1: RECOMMENDED values for delay based controller

Table 1

5. Loss-based control

A second part of the congestion controller bases its decisions on the round-trip time, packet loss and available bandwidth estimates A_hat received from the delay-based controller. The available bandwidth

estimates computed by the loss-based controller are denoted with As_hat.

The available bandwidth estimates A_hat produced by the delay-based controller are only reliable when the size of the queues along the path sufficiently large. If the queues are very short, over-use will only be visible through packet losses, which are not used by the delay-based controller.

The loss-based controller SHOULD run every time feedback from the receiver is received.

- If 2-10% of the packets have been lost since the previous report from the receiver, the sender available bandwidth estimate As_hat(i) will be kept unchanged.

- If more than 10% of the packets have been lost a new estimate is calculated as As_hat(i) = As_hat(i-1)(1-0.5p), where p is the loss ratio.

- As long as less than 2% of the packets have been lost As_hat(i) will be increased as As_hat(i) = 1.05(As_hat(i-1))

The new bandwidth estimate is lower-bounded by the TCP Friendly Rate Control formula [RFC3448] and upper-bounded by the delay-based estimate of the available bandwidth A_hat(i), where the delay-based estimate has precedence:

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

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

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

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

In words: The loss-based estimate will never be larger than the delay-based estimate, and will never be lower than the estimate from the TFRC formula except if the delay-based estimate is lower than the TFRC estimate.

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 bitrate. Therefore we will soon enough reach above the 10% threshold and adjust As_hat(i). However, if the packet loss ratio does not increase, the losses are probably not related to self-inflicted congestion and therefore we should not react on them.

6. Interoperability Considerations

In case a sender implementing these algorithms talks to a receiver which do not implement any of the proposed RTCP messages and RTP header extensions, it is suggested that the sender monitors RTCP receiver reports and uses the fraction of lost packets and the round-trip time as input to the loss-based controller. The delay-based controller should be left disabled.

7. Implementation Experience

This algorithm has been implemented in the open-source WebRTC project, has been in use in Chrome since M23, and is being used by Google Hangouts.

Deployment of the algorithm have revealed problems related to, e.g, congested or otherwise problematic WiFi networks, which have led to algorithm improvements. The algorithm has also been tested in a multi-party conference scenario with a conference server which terminates the congestion control between endpoints. This ensures that no assumptions are being made by the congestion control about maximum send and receive bitrates, etc., which typically is out of control for a conference server.

8. Further Work

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

Work that can be done on this basis includes:

- Considerations of integrated loss control: How loss and delay control can be better integrated, and the loss control improved.
- Considerations of locus of control: evaluate the performance of having all congestion control logic at the sender, compared to splitting logic between sender and receiver.
- Considerations of utilizing ECN as a signal for congestion estimation and link over-use detection.

9. IANA Considerations

This document makes no request of IANA.

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

10. Security Considerations

An attacker with the ability to insert or remove messages on the connection would have the ability to disrupt rate control. This could make the algorithm to produce either a sending rate under-utilizing the bottleneck link capacity, or a too high sending rate causing network 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.

11. 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.

12. References

12.1. 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.holmer-rmcat-transport-wide-cc-extensions]
Holmer, S., Flodman, M., and E. Sprang, "RTP Extensions for Transport-wide Congestion Control", draft-holmer-rmcat-transport-wide-cc-extensions-00 (work in progress), March 2015.

[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.

[abs-send-time] "RTP Header Extension for Absolute Sender Time", <http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time>.

12.2. Informative References

[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

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

A.2. Version -01 to -02

- Defined the term "frame", incorporating the transmission time offset into its definition, and removed references to "video frame".

- Referred to "m(i)" from the text to make the derivation clearer.
- Made it clearer that we modify our estimates of available bandwidth, and not the true available bandwidth.
- Removed the appendixes outlining new extensions, added pointers to REMB draft and RFC 5450.

A.3. Version -02 to -03

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

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

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

A.7. rmcat -02 to -03

- Swapped receiver-side/sender-side controller with delay-based/ loss-based controller as there is no longer a requirement to run the delay-based controller on the receiver-side.
- Removed the discussion about multiple streams and transmission time offsets.
- Introduced a new section about "Feedback and extensions".
- Improvements to the threshold adaptation in the "Over-use detector" section.

- Swapped the previous MIMD rate control algorithm for a new AIMD rate control algorithm.

Authors' Addresses

Stefan Holmer
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: holmer@google.com

Henrik Lundin
Google
Kungsbron 2
Stockholm 11122
Sweden

Gaetano Carlucci
Politecnico di Bari
Via Orabona, 4
Bari 70125
Italy

Email: gaetano.carlucci@poliba.it

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

J. Fu
X. Zhu
M. Ramalho
W. Tan
Cisco Systems

July 6, 2015

Evaluation Test Cases for Interactive Real-Time Media over Wi-Fi Networks
draft-fu-rmcat-wifi-test-case-01

Abstract

An increasing proportion of multimedia communication applications, including real-time interactive voice and video, are transported over Wi-Fi networks (i.e., wireless local area networks following IEEE 802.11 standards) today. It is therefore important to evaluate candidate congestion control schemes designed in the RMCAT Working Group over test cases that include Wi-Fi access links. This draft serves such a purpose, and is complementary to [I-D.ietf-rmcat-eval-test] and [I-D.draft-sarker-rmcat-cellular-eval-test-cases]

Table of Contents

1. Introduction

Given the prevalence of Internet access links over Wi-Fi, it is important to evaluate candidate RMCAT congestion control solutions over Wi-Fi test cases. Such evaluations should also highlight the inherent different characteristics of Wi-Fi networks in contrast to Wired networks:

- The wireless radio channel is subject to interference from nearby transmitters, multipath fading, and shadowing, causing

fluctuations in link throughput and sometimes an error-prone communication environment

- Available network bandwidth is not only shared over the air between cocurrent users, but also between uplink and downlink traffic due to the half duplex nature of wireless transmission medium.

- Packet transmessions over Wi-Fi are susceptible to contentions and collisions over the air. Consequently, traffic load beyond a certain utilization level over a Wi-Fi network can introduce frequent collisions and significant network overhead. This, in turn, leads to excessive delay, retransmission, loss and lower effective bandwidth for applications.

- The IEEE 802.11 standard (i.e., Wi-Fi) supports multi-rate transmission capabilities by dynamically choosing the most appropriate modulation scheme for a given received singal strength. A different choice of Physical-layer rate will lead to different application-layer throughput.

- Presence of legancy 802.11b networks can significantly slow down the the rest of a modern Wi-Fi Network, since it takes longer to transmit the same packet over a slower link than over a faster link. [Editor’s note: maybe include a reference here instead.]

- Handover from one Wi-Fi Access Point (AP) to another may cause packet delay and loss.

- IEEE 802.11e defined EDCA/WMM (Enhanced DCF Channel Access/Wi-Fi Multi-Media) to give voice and video streams higher priority over pure data applications (e.g., file transfers).

As we can see here, presence of Wi-Fi network in different different network topologies and traffic arrival can exert different impact on the network performance in terms of video transport rate, packet loss and delay that, in turn, effect end-to-end real-time multimedia congestion control.

Currently, the most widely used IEEE 802.11 standards are 802.11g and 802.11n. The industry is moving towards 802.11ac and, potentially, 802.11ad. IEEE 802.11b is legency standard, and 802.11a has not been widely adopted. Throughout this draft, unless otherwise mentioned, test cases are described using 802.11g mostly due to its wide availability both in test equipments and network simulation platform. Whenever possible, it is recommended to extend some of the experiments to 802.11ac so as to reflect a more mordent Wi-Fi network setting.

Since Wi-Fi network normally connects to a wired infrastructure, either the wired network or the Wi-Fi network could be the bottleneck. In the following section, we describe basic test cases for both scenarios separately.

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 RFC2119 [RFC2119].

3. Test Scenarios with Wired Bottlenecks

The test scenarios below are intended to mimic the set up of video conferencing over Wi-Fi connections from the home. Typically, the Wi-Fi home network is not congested and the bottleneck is present over the wired home access link. Although it is expected that test evaluation results from this section are similar to those from the all-wired test cases (draft-sarker-rmcat-eval-test), it is worthwhile to run through these tests as sanity checks.

3.1. Single RMCAT Flow over Wired Bottleneck

This test case is designed to measure the responsiveness of the candidate algorithm when the Wi-Fi hop of the connection is uncongested.

Figure 1 illustrates topology of this test.

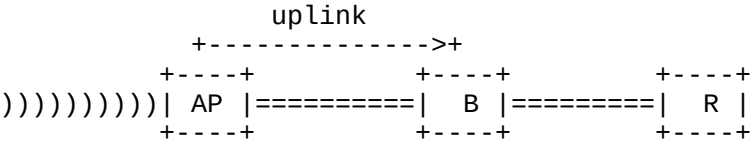


Figure 1: Single Flow over Wired Bottleneck

Testbed attributes:

- Test duration: 100s
- Path characteristics:
  - Uplink capacity: 1Mbps
  - One-Way propagation delay: 50ms.

- Maximum end-to-end jitter: 30ms
- Bottleneck queue type: Drop tail.
- Bottleneck queue size: 300ms.
- Path loss ratio: 0%.

- Application-related:
  - Media Traffic:
    - Media type: Video
    - Media direction: forward.
    - Number of media sources: One (1)
    - Media timeline:
      - Start time: 0s.
      - End time: 99s.
  - Competing traffic:
    - Number of sources : Zero (0)

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. 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.

3.2. Bidirectional RMCAT Flow over Wired Bottleneck

This test case is designed to evaluate performance of the candidate algorithm when lack of enough feedback.

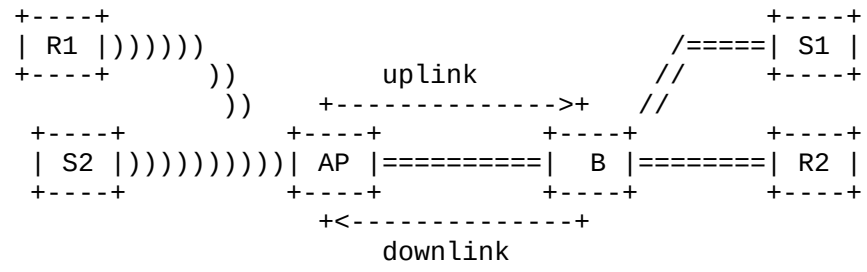


Figure 2: One Bi-directional Flow over Wired Bottleneck

Testbed attributes:

- Test duration: 100s
- Path characteristics:
  - uplink capacity: 1Mbps
  - One-Way propagation delay: 50ms.
  - Maximum end-to-end jitter: 30ms
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue size: 300ms.
  - Path loss ratio: 0%.
- Application characteristics:
  - Media Traffic:
    - Media type: Video
    - Media direction: forward and backward.
    - Number of media sources: Two (2)
    - Media timeline:
      - Start time: 0s.
      - End time: 99s.

* Competing traffic:

  + Number and Types of sources : zero (0)

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

3.3. RMCAT Flow Competing with Long TCP over Wired Bottleneck

This test case is designed to measure the performance of the candidate algorithm when lack of enough feedback.

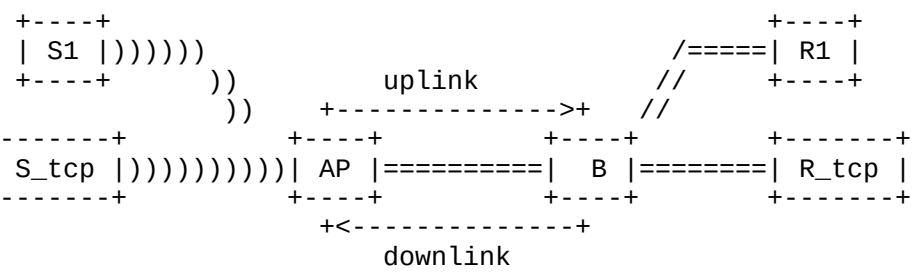


Figure 3: RMCAT vs. TCP over Wired Bottleneck

Testbed attributes:

Test duratiion: 100s

Path characteristics:

* uplink capacity: 1Mbps

* One-Way propagation delay: 50ms.

* Maximum end-to-end jitter: 30ms

* Bottleneck queue type: Drop tail.

* Bottleneck queue size: 300ms.

* Path loss ratio: 0%.

Application-related:

* Media Traffic:
  + Media type: Video
  + Media direction: forward.
  + Number of media sources: One (1)
  + Media timeline:
    - Start time: 0s.
    - End time: 99s.
* Competing traffic:
  + Types of sources : long-lived TCP
  + Number of sources: One (1)
  + Traffic direction : forward
  + Congestion control: Default TCP congestion control [TBD].
  + Traffic timeline:
    - Start time: 0s.
    - End time: 119s.

Expected behavior: the candidate algorithm should be able to avoid congestion collapse, and get fair share of the bandwidth. In the worst case, the media stream will fall to the minimum media bit rate.

4. Bottleneck over Wireless Network

These test cases assume that the wired portion along the media path are well-provisioned. The bottleneck is in the Wi-Fi network over wireless. This is to mimic the enterprise/coffee-house scenarios.

4.1. Adaptive rate selection with single RMCAT flow

Since morden IEEE 802.11 standards supports far higher data rates than the maximum requirements of individual RMCAT flows, in this test the legacy standard 802.11b is chosen to test the single RMCAT flow case. 802.11b Adaptive rate selection can operate at 11 Mbps in terms of PHY-layer transmission rate, and falls back to 5.5 Mbps, 2 Mbps, and 1 Mbps when the wireless client moves away from the access point.

[Editor's Note: we may want to move this section to Section 5.4 instead.]

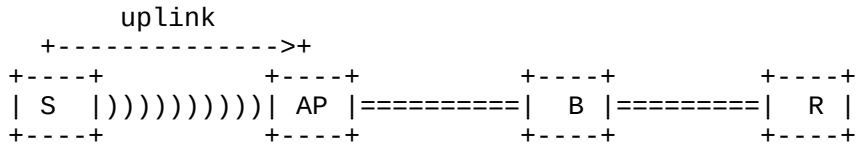


Figure 4: One RMCAT Flow over Wireless Bottleneck

Testbed attributes:

Test duratiion: 100s

Path characteristics:

* Wired path capacity: 100Mbps
* Wi-Fi PHY Rate: 1Mbps (PHY rate)
* One-Way propagation delay: 50ms.
* Maximum end-to-end jitter: 30ms
* Bottleneck queue type: Drop tail.
* Bottleneck queue size: 300ms.
* Path loss ratio: 0%.

Application characteristics:

* Media Traffic:
  + Media type: Video
  + Media direction: forward and backward.
  + Number of media sources: One (1)
  + Media timeline:

- Start time: 0s.

- End time: 99s.

* Competing traffic:

  + Number and Types of sources : zero (0)

Test Specific Information:

* This test will change the distance between station and AP (need some experiment), and incur the adaptive rate selection variation as listed in Figure 5.

Variation pattern index	Path direction	Start time	PHY-layer rate
One	Forward	0s	5.5 Mbps
Two	Forward	40s	2 Mbps
Three	Forward	60s	1 Mbps
Four	Forward	80s	2 Mbps

Figure 5: Adaptive rate variation pattern for uplink direction

Expected behavior: The rate adaptation algorithm run at application level should follow the adaptation in 802.11 mac layer.

4.2. Multiple RMCAT Flows Sharing the Wireless Downlink

This test case is for studying the impact of contention on competing RMCAT flows. Specifications for IEEE 802.11g with a physical-layer transmission rate of 54 Mbps is chosen. Not that retransmission and MAC-layer headers and control packets may be sent at a lower link speed. The total application-layer throughput (reasonable distance, low interference and small number of contention stations) for 802.11g is around 20 Mbps. Consequently, a total of 16 RMCAT flows are needed for saturating the wireless interface in this experiment.

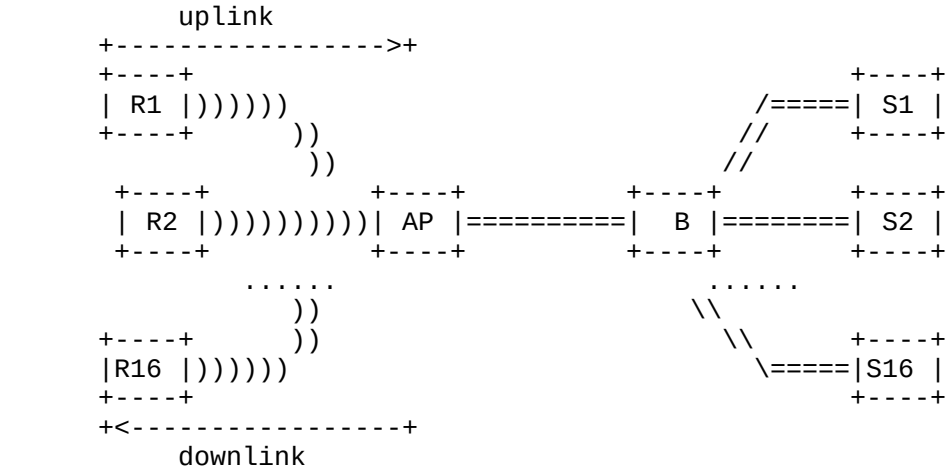


Figure 6: Multiple RMCAT Flows Sharing the Wireless Downlink

Testbed attributes:

- Test duratiion: 100s
- Path characteristics:
  - Wired path capacity: 100Mbps
  - Wi-Fi PHY Rate: 54Mbps (PHY rate)
  - One-Way propagation delay: 50ms.
  - Maximum end-to-end jitter: 30ms
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue size: 300ms.
  - Path loss ratio: 0%.
- Application characteristics:
  - Media Traffic:
    - Media type: Video
    - Media direction: backward.

+ Number of media sources: Sixteen (16)

+ Media timeline:

- Start time: 0s.

- End time: 99s.

* Competing traffic:

+ Number and Types of sources : Zero (0)

Expected behavior: All RMCAT flow should get fair share of the bandwidth. Overall bandwidth usage should be no less than same case with TCP flows (using TCP as performance benchmark). The delay and loss should be within acceptable range for real-time multimedia flow.

4.3. Multiple RMCAT Flows Sharing the Wireless Uplink

This test case is different with the previous section with mostly downlink transmissions. When multiple clients attempt to transmit video packets uplink over the wireless interface, they introduce more frequent contentions and potentially collisions. As a results, the per-client throught is expected to be lower than the downlink-only scenario.

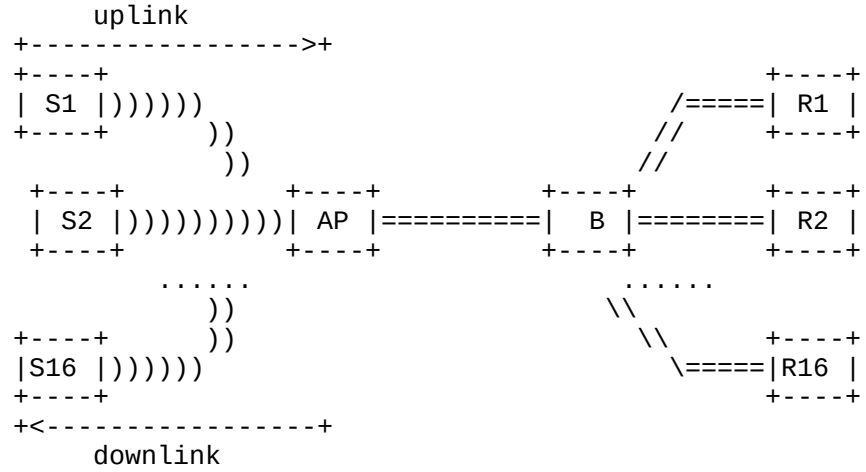


Figure 7: Multiple RMCAT Flows Sharing the Wireless Uplink

Testbed attributes:

- Test duratiion: 100s
- Path characteristics:
  - Wired path capacity: 100Mbps
  - Wi-Fi PHY Rate: 54Mbps (PHY rate)
  - Maximum end-to-end jitter: 30ms
  - One-Way propagation delay: 50ms.
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue size: 300ms.
  - Path loss ratio: 0%.
- Application characteristics:
  - Media Traffic:
    - Media type: Video
    - Media direction: forward and backward.
    - Number of media sources: Sixteen (16)
    - Media timeline:
      - Start time: 0s.
      - End time: 99s.
  - Competing traffic:
    - Number and Types of sources : Zero (0)

Expected behavior: All RMCAT flow should get fair share of the bandwidth, and the overall bandwidth usage should no less than same case with TCP flows (use TCP as performance benchmark). The delay and loss should be in acceptable range for real-time multimedia flow (might need rtp circuit breaker to guarantee that?).

4.4. Multiple Bi-Directional RMCAT Flows Sharing the Wireless Bottleneck

This one differs with previous contention cases because Wi-Fi share bandwdith between uplink and downlink.

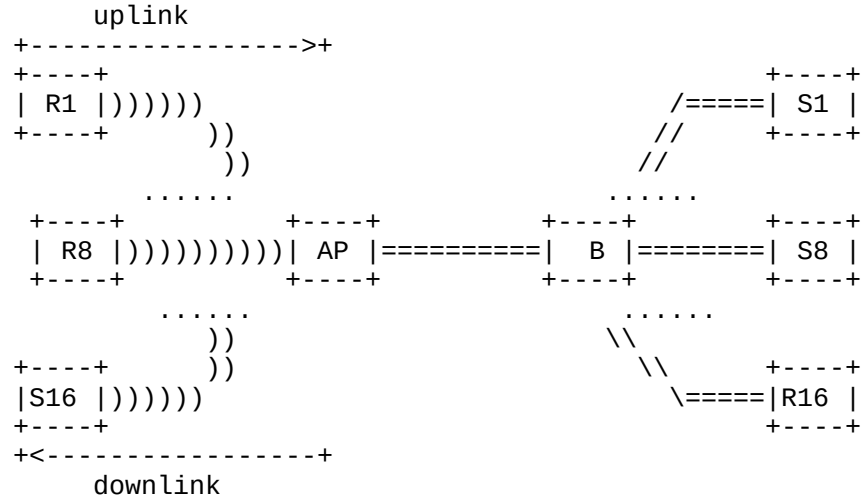


Figure 8: Multiple Bi-Directional RMCAT Flows Sharing the Wireless Bottleneck

Testbed attributes:

- Test duratiion: 100s
- Path characteristics:
  - Wired path capacity: 100Mbps
  - Wi-Fi PHY Rate: 54Mbps (PHY rate)
  - One-Way propagation delay: 50ms.
  - Maximum end-to-end jitter: 30ms
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue size: 300ms.

    * Path loss ratio: 0%.

  o Application characteristics:

    * Media Traffic:

      + Media type: Video

      + Media direction: forward and backward.

      + Number of media sources: Sixteen (16), 8 for uplink, 8 for downlink.

      + Media timeline:

        - Start time: 0s.

        - End time: 99s.

    * Competing traffic:

      + Number and Types of sources : zero (0)

  Expected behavior: All (uplink/downlink) RMCAT flow should get fair share of the bandwidth, and the overall bandwidth usage should no less than same case with TCP flows (use TCP as performance benchmark). The delay and loss should be in acceptable range for real-time multimedia flow (might need rtp circuit breaker to guarantee that?).

4.5. Multiple RMCAT and TCP Flows Sharing the Wireless Uplink

  This case having both long lived TCP and RMCAT sharing the uplink at the same time. This is for testing how RMCAT competing with long lived TCP flow in a congested Wi-Fi network.

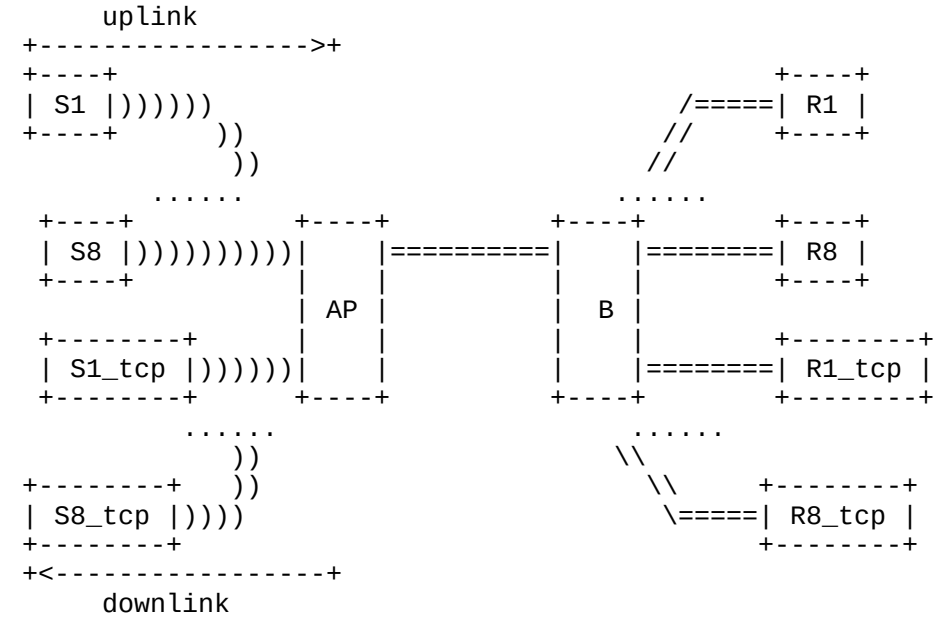


Figure 9: Multiple RMCAT and TCP Flows Sharing the Wireless Uplink

Testbed attributes:

- Test duratiion: 100s
- Path characteristics:
  - Wired path capacity: 100Mbps
  - Wi-Fi PHY Rate: 54Mbps (PHY rate)
  - One-Way propagation delay: 50ms.
  - Maximum end-to-end jitter: 30ms
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue size: 300ms.
  - Path loss ratio: 0%.
- Application characteristics:

* Media Traffic:
  + Media type: Video
  + Media direction: forward.
  + Number of media sources: Eigth (8).
  + Media timeline:
    - Start time: 0s.
    - End time: 99s.
* Competing traffic:
  + Type of sources: long-live TCP.
  + Number of sources : Eight (8)
  + Traffic direction : forward
  + Congestion control: Default TCP congestion control [TBD].
  + Traffic timeline:
    - Start time: 0s.
    - End time: 99s.

Expected behavior: All RMCAT flows should get comparable share of the network bandwidth with respect to competing TCP flows. The overall bandwidth usage should no less than same case with TCP flows (use TCP as performance benchmark). The delay and loss should be in acceptable range for real-time multimedia flow (might need rtp circuit breaker to guarantee that?).

4.6. Multiple RMCAT and TCP Flows Sharing the Wireless Downlink

This case having both long lived TCP and RMCAT on the downlink at the same time. This is for testing how RMCAT competing with long lived TCP flow in crowed Wi-Fi network. This differs from test scenario in the previous section becauase less contention on the Wi-Fi network because most media data is sent from AP to stations.

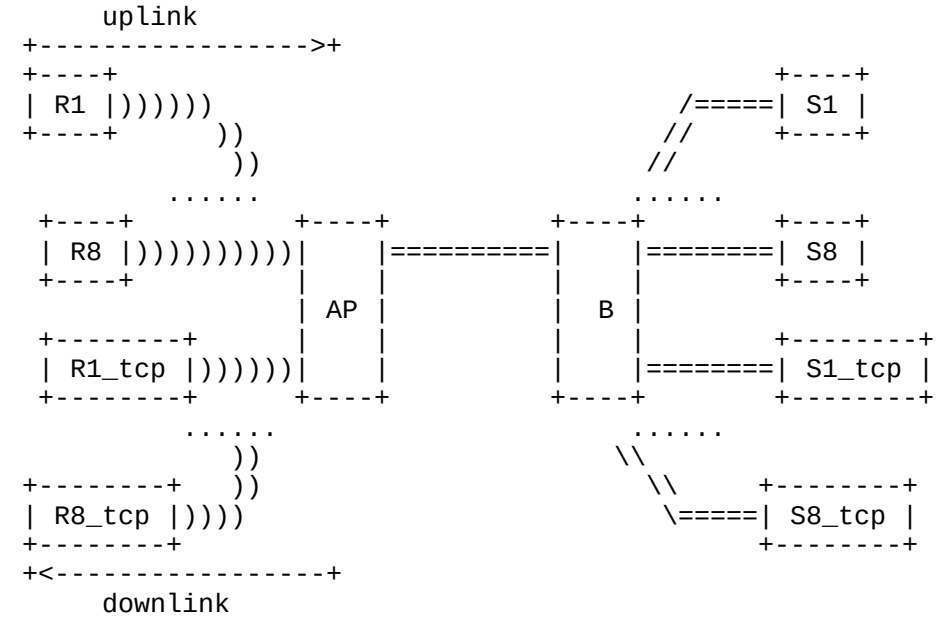


Figure 10: Multiple RMCAT and TCP Flows Sharing the Wireless Downlink

Testbed attributes:

- Test duratiion: 100s
- Path characteristics:
  - Wired path capacity: 100Mbps
  - Wi-Fi PHY Rate: 54Mbps (PHY rate)
  - One-Way propagation delay: 50ms.
  - Maximum end-to-end jitter: 30ms
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue depth: 300ms.
  - Path loss ratio: 0%.
- Application characteristics:

* Media Traffic:
  + Media type: Video
  + Media direction: backward.
  + Number of media sources: Eight (8).
  + Media timeline:
    - Start time: 0s.
    - End time: 99s.
* Competing traffic:
  + Number of sources: Eight (8).
  + Types of sources : long-lived TCP.
  + Traffic direction : forward
  + Congestion control: Default TCP congestion control.
  + Traffic timeline:
    - Start time: 0s.
    - End time: 99s.

Expected behavior: All RMCAT flows should get comparable share of the network bandwidth with respect to competing TCP flows. The overall bandwidth usage should no less than same case with TCP flows (use TCP as performance benchmark). The delay and loss should be in acceptable range for real-time multimedia flow (might need rtp circuit breaker to guarantee that?).

4.7. Multiple Bi-Directional RMCAT and TCP Flows Sharing the Wireless Bottleneck

This case having both long lived TCP and RMCAT on the both direction at the same time. This is for testing how RMCAT competing with long lived TCP flow in a congested Wi-Fi network. This differs from previouss cases as both uplink and downlink flows share the same wireless bottleneck.

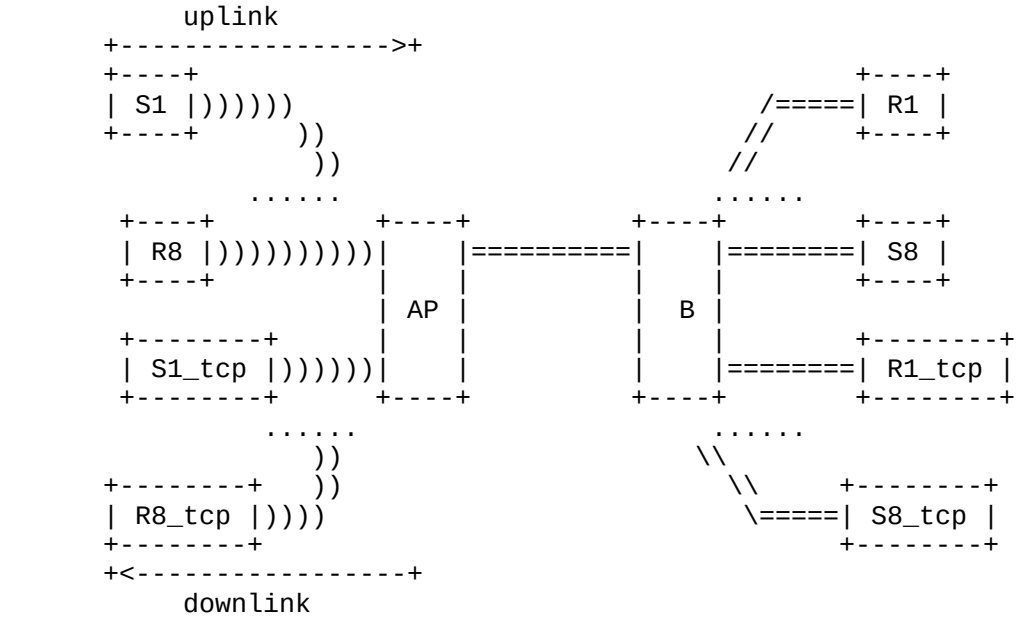


Figure 11: Multiple Bi-Directional RMCAT and TCP Flows Sharing the Wireless Bottleneck

Testbed attributes:

- Test duratiion: 100s
- Path characteristics:
  - Wired path capacity: 100Mbps
  - Wi-Fi PHY Rate: 54Mbps (PHY rate)
  - One-Way propagation delay: 50ms.
  - Maximum end-to-end jitter: 30ms
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue size: 300ms.
  - Path loss ratio: 0%.
- Application characteristics:

* Media Traffic:
  + Media type: Video
  + Media direction: forward and backward.
  + Number of media sources: Eight (8). Four (4) forward, Four (4) backward
  + Media timeline:
    - Start time: 0s.
    - End time: 99s.
* Competing traffic:
  + Number of sources : Eight (8). Four (4) forward, Four (4) backward.
  + Type of sources: long-live TCP.
  + Traffic direction : forward and backward.
  + Congestion control: Default TCP congestion control.
  + Traffic timeline:
    - Start time: 0s.
    - End time: 99s.

Expected behavior: All RMCAT flows should get comparable share of the network bandwidth with respect to competing TCP flows. The overall bandwidth usage should no less than same case with TCP flows (use TCP as performance benchmark). The delay and loss should be in acceptable range for real-time multimedia flow (might need rtp circuit breaker to guarantee that?).

5. Other potential test cases

5.1. Wi-Fi Roaming

Wi-Fi roaming need scanning, authentication and re-association which will cause packet drops and delay, and interrupt the network connection. RMCAT congestion control algorithms should at least recover (if affected by the transition process) after roaming.

5.2. Wi-Fi/Cellular Switch

The phone can switch automatically between Wi-Fi and Cellular network when the other is not available, and some phones like "Samsung Galaxy" have smart network switch to switching to network has better connectivity automatically. Unlike Wi-Fi Roaming, such kind of switch might or might not interrupt the network connection, and might change the route. RMCAT congestion control should be able to cope with the changes.

5.3. EDCA/WMM usage

EDCA/WMM is prioritized QoS with four traffic classes (or Access Categories) with differing priorities. RMCAT flow should have better performance (lower delay, less loss) with EDCA/WMM enabled when competing against non-interactive background traffic (e.g., file transfers). When most of the traffic over Wi-Fi is dominated by media, however, turning on WMM may actually degrade performance. This is a topic worthy of further investigation.

5.4. Legacy 802.11b Effects

When there is 802.11b devices connected to modern 802.11 network, it may affect the performance of the whole network. Additional test cases can be added to evaluate the affects of legancy devices on the performance of RMCAT congestion control algorithm.

6. IANA Considerations

There are no IANA impacts in this memo.

7. References

7.1. Normative References

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

[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-01 (work in progress), March 2014.

[I-D.ietf-rmcat-eval-test]
Sarker, Z., Singh, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-ietf-rmcat-eval-test-01 (work in progress), March 2015.

[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.draft-sarker-rmcat-cellular-eval-test-cases]
Sarker, Z., "Evaluation Test Cases for Interactive Real-Time Media over Cellular Networks", <https://tools.ietf.org/html/draft-sarker-rmcat-cellular-eval-test-cases-02>.

7.2. 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.

Authors' Addresses

Jiantao Fu
Cisco Systems
707 Tasman Drive
Milpitas, CA 95035
USA

Email: jianfu@cisco.com

Xiaoqing Zhu
Cisco Systems
12515 Research Blvd., Building 4
Austin, TX 78759
USA

Email: xiaoqzhu@cisco.com

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

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

Wei-Tian Tan
Cisco Systems
725 Alder Drive
Milpitas, CA 95035
USA

Email: dtan2@cisco.com

Network Working Group
Internet Draft
Intended Status: Informational
Expires: October 29, 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

April 28, 2015

NADA: A Unified Congestion Control Scheme for Real-Time Media
draft-ietf-rmcat-nada-00

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/1id-abstracts.html

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

Table of Contents

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:

$$z_n = t_{r,n} - t_{s,n},$$

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 z_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 z_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:

$$d_n = z_n - d_f.$$

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/marking 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:

$$\tilde{d}_n = \hat{d}_n, \quad \text{if } \hat{d}_n < d_{th};$$

$$\tilde{d}_n = d_{th} \frac{(d_{max} - \hat{d}_n)^4}{(d_{max} - d_{th})^4}, \quad \text{if } d_{th} < \hat{d}_n < d_{max};$$

$$\tilde{d}_n = 0, \quad \text{if } \hat{d}_n > d_{max}.$$

Here, the queuing delay value is unchanged when it is below the first threshold d_th; it is discounted following a non-linear curve when its value falls between d_th and d_max; above d_max, the high queuing delay value no longer counts toward congestion control.

When queuing delay is in the range (0, d_th), NADA operates in pure delay-based mode if no losses/markings are present. When queuing delay exceeds d_max, NADA reacts to loss/marking only. In between d_th and d_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_th is chosen as 50ms and d_max is chosen as 400ms. The impact of the choice of d_th and d_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:

$$x\_n = d\_tilde\_n + p\_M*d\_M + p\_L*d\_L, \quad (1)$$

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 <d_hat_n, x_n, x’_n, R_r> in RTCP feedback messages to aid the sender in its calculation of target rate. The queuing delay value d_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:

$$x'\_n = \frac{x\_n - x\_(n-k)}{delta}, \quad (2)$$

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 feed 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.

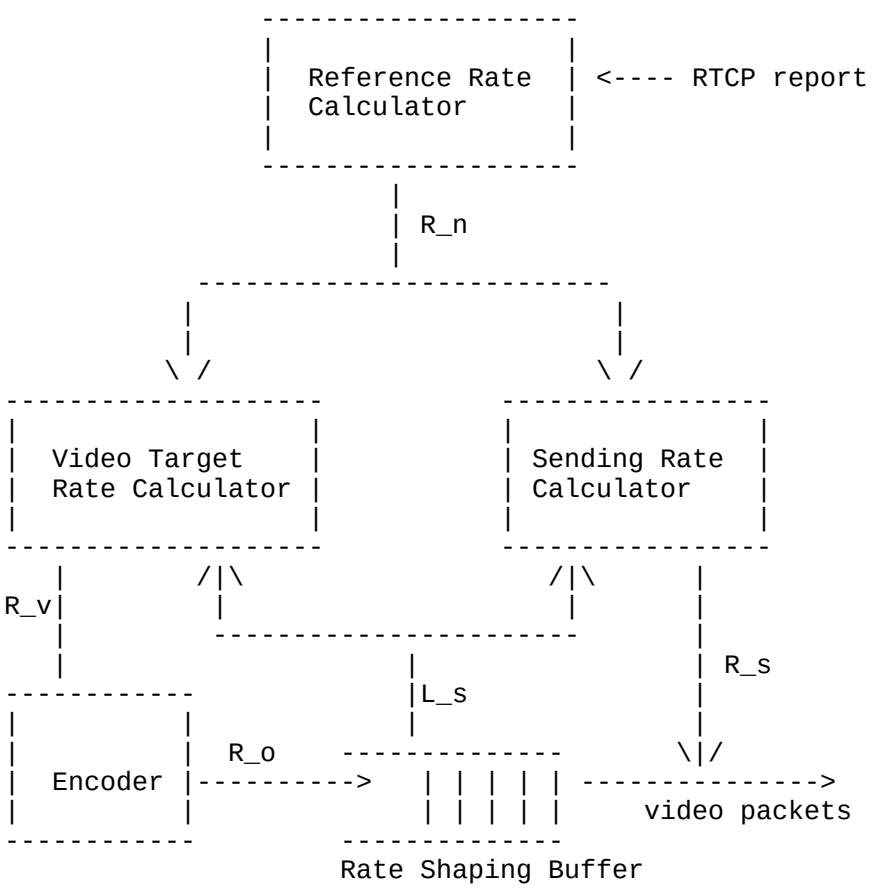


Figure 1 NADA Sender Structure

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_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_th milliseconds of queuing delay at the bottleneck during the ramp-up process. A typical value of T_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:

$$\text{gamma} = \min\left[\text{gamma\_0}, \frac{\text{T\_th}}{\text{RTT\_0+delta\_0}}\right] \quad (3)$$

$$\text{R\_n} = (1+\text{gamma})\ \text{R\_r} \quad (4)$$

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 un-congested.

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:

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

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:

$$R_v = R_n - \beta_v * \frac{L_s}{\tau_v}. \quad (8)$$

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

$$R_s = R_n + \beta_s * \frac{L_s}{\tau_v}. \quad (9)$$

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 (LEDBAT)", 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 Evalua6on 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 Interium, 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:

$$p = 0, \text{ if } q < q\_lo;$$

$$p = p\_max * \frac{q\_avg - q\_lo}{q\_hi - q\_lo}, \text{ if } q\_lo <= q < q\_hi;$$

$$p = 1, \text{ if } q >= q\_hi.$$

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;

$$p = p\_max * \frac{b - b\_tk - b\_lo}{b\_hi - b\_lo}, \text{ if } b\_lo <= b - b\_tk < b\_hi;$$

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*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

Charles Ganzhorn
7900 International Drive
International Plaza, Suite 400
Bloomington, MN 55425, USA
Email: charles.ganzhorn@gmail.com

Paul E. Jones
7025 Kit Creek Rd.
Research Triangle Park, NC 27709, USA
Email: 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

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: August 7, 2019

X. Zhu
R. Pan
M. Ramalho
S. Mena
P. Jones
J. Fu
Cisco Systems
S. D'Aronco
EPFL

February 3, 2019

NADA: A Unified Congestion Control Scheme for Real-Time Media
draft-ietf-rmcat-nada-10

Abstract

This document describes NADA (network-assisted dynamic adaptation), a novel congestion control scheme 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.

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 https://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 7, 2019.

Table of Contents

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. The requirements for the congestion control algorithm are outlined in [I-D.ietf-rmcat-cc-requirements].

This document describes an experimental congestion control 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. Such a unified sender behavior distinguishes NADA from other congestion control schemes for real-time media. In addition, its core congestion control algorithm is designed to guarantee stability for path round-trip-times (RTTs) below a prescribed bound (e.g., 250ms with default parameter choices). It further supports weighted bandwidth sharing among competing video flows with different priorities. The signaling mechanism consists of standard RTP timestamp [RFC3550] and RTCP feedback reports. The definition of standardized RTCP feedback message requires future work so as to support the successful operation of several congestion control schemes for real-time interactive media.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. System Overview

Figure 1 shows the end-to-end system for real-time media transport that NADA operates in. Note that there also exist network nodes along the reverse (potentially uncongested) path that the RTCP

feedback reports traverse. Those network nodes are not shown in the figure for sake of abrevity.

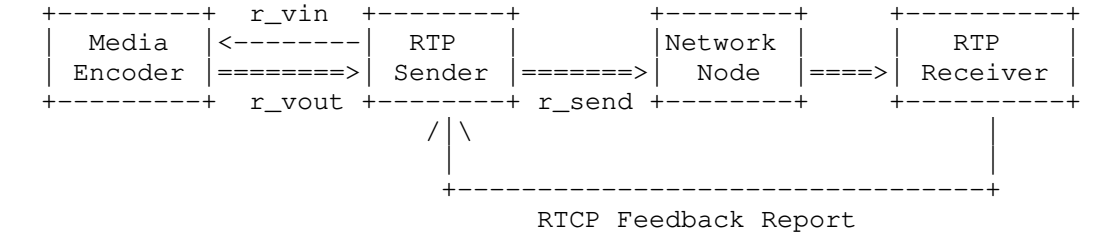


Figure 1: System Overview

- Media encoder with rate control capabilities. It encodes raw media (audio and video) frames into compressed bitstream which is later packetized into RTP packets. As discussed in [I-D.ietf-rmcat-video-traffic-model], the actual output rate from the encoder r_vout may fluctuate around the target r_vin. Furthermore, it is possible that the encoder can only react to bit rate changes at rather coarse time intervals, e.g., once every 0.5 seconds.

- RTP sender: responsible for calculating the NADA reference rate 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_vin, and for regulating the actual sending rate r_send accordingly. The RTP sender also generates a sending timestamp for each outgoing packet.

- RTP receiver: responsible for measuring and estimating end-to-end delay (based on sender timestamp), packet loss (based on RTP sequence number), ECN marking ratios (based on [RFC6679]), and receiving rate (r_recv) of the flow. It calculates the aggregated congestion signal (x_curr) that accounts for queuing delay, ECN markings, and packet losses. The receiver also determines the mode for sender rate adaptation (rmode) based on whether the flow has encountered any standing non-zero congestion. The receiver sends periodic RTCP reports back to the sender, containing values of x_curr, rmode, and r_recv.

- 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 PIE, FQ-CoDel, RED-based ECN marking, and PCN marking using a token bucket algorithm. Note that network node operation is out of control for the design of NADA.

4. Core Congestion Control Algorithm

Like TCP-Friendly Rate Control (TFRC) [Floyd-CCR00] [RFC5348], NADA is a rate-based congestion control algorithm. In its simplest form, the sender reacts to the collection of network congestion indicators in the form of an aggregated congestion signal, and operates in one of two modes:

- Accelerated ramp-up: when the bottleneck is deemed to be underutilized, the rate increases multiplicatively with respect to the rate of previously successful transmissions. The rate increase mutliplier (gamma) is calculated based on observed round-trip-time and target feedback interval, so as to limit self-inflicted queuing delay.

- Gradual rate update: in the presence of non-zero aggregate congestion signal, the sending rate is adjusted in reaction to both its value (x_curr) and its change in value (x_diff).

This section introduces the list of mathematical notations and describes the core congestion control algorithm at the sender and receiver, respectively. Additional details on recommended practical implementations are described in Section 5.1 and Section 5.2.

4.1. Mathematical Notations

This section summarizes the list of variables and parameters used in the NADA algorithm. Figure 3 also includes the default values for choosing the algorithm parameters either to represent a typical setting in practical applications or based on theoretical and simulation studies. See Section 6.3 for some of the discussions on the impact of parameter values. In the experimental phase of this draft, additional studies in real-world settings will gather further learnings on how to choose and adapt these parameter values in practical deployment.

Notation	Variable Name
t_curr	Current timestamp
t_last	Last time sending/receiving a feedback
delta	Observed interval between current and previous feedback reports: delta = t_curr-t_last
r_ref	Reference rate based on network congestion
r_send	Sending rate
r_recv	Receiving rate
r_vin	Target rate for video encoder
r_vout	Output rate from video encoder
d_base	Estimated baseline delay
d_fwd	Measured and filtered one-way delay
d_queue	Estimated queuing delay
d_tilde	Equivalent delay after non-linear warping
p_mark	Estimated packet ECN marking ratio
p_loss	Estimated packet loss ratio
x_curr	Aggregate congestion signal
x_prev	Previous value of aggregate congestion signal
x_diff	Change in aggregate congestion signal w.r.t. its previous value: x_diff = x_curr - x_prev
rmode	Rate update mode: (0 = accelerated ramp-up; 1 = gradual update)
gamma	Rate increase multiplier in accelerated ramp-up mode
loss_int	Measured average loss interval in packet count
loss_exp	Threshold value for setting the last observed packet loss to expiration
rtt	Estimated round-trip-time at sender
buffer_len	Rate shaping buffer occupancy measured in bytes

Figure 2: List of variables.

Notation	Parameter Name	Default Value
PRIO	Weight of priority of the flow	1.0
RMIN	Minimum rate of application supported by media encoder	150 Kbps
RMAX	Maximum rate of application supported by media encoder	1.5 Mbps
XREF	Reference congestion level	10ms
KAPPA	Scaling parameter for gradual rate update calculation	0.5
ETA	Scaling parameter for gradual	2.0

	rate update calculation	
TAU	Upper bound of RTT in gradual rate update calculation	500ms
DELTA	Target feedback interval	100ms
LOGWIN	Observation window in time for calculating packet summary statistics at receiver	500ms
QEPS	Threshold for determining queuing delay build up at receiver	10ms
DFILT	Bound on filtering delay	120ms
GAMMA_MAX	Upper bound on rate increase ratio for accelerated ramp-up	0.5
QBOUND	Upper bound on self-inflicted queuing delay during ramp up	50ms
MULTILOSS	Multiplier for self-scaling the expiration threshold of the last observed loss (loss_exp) based on measured average loss interval (loss_int)	7.
QTH	Delay threshold for invoking non-linear warping	50ms
LAMBDA	Scaling parameter in the exponent of non-linear warping	0.5
PLRREF	Reference packet loss ratio	0.01
PMRREF	Reference packet marking ratio	0.01
DLOSS	Reference delay penalty for loss when packet loss ratio is at PLRREF	10ms
DMARK	Reference delay penalty for ECN marking when packet marking is at PMRREF	2ms
FPS	Frame rate of incoming video	30
BETA_S	Scaling parameter for modulating outgoing sending rate	0.1
BETA_V	Scaling parameter for modulating video encoder target rate	0.1
ALPHA	Smoothing factor in exponential smoothing of packet loss and marking ratios	0.1

Figure 3: List of algorithm parameters and their default values.

4.2. Receiver-Side Algorithm

The receiver-side algorithm can be outlined as below:

```
On initialization:
  set d_base = +INFINITY
  set p_loss = 0
  set p_mark = 0
  set r_recv = 0
  set both t_last and t_curr as current time in milliseconds

On receiving a media packet:
  obtain current timestamp t_curr from system clock
  obtain from packet header sending time stamp t_sent
  obtain one-way delay measurement: d_fwd = t_curr - t_sent
  update baseline delay: d_base = min(d_base, d_fwd)
  update queuing delay:  d_queue = d_fwd - d_base
  update packet loss ratio estimate p_loss
  update packet marking ratio estimate p_mark
  update measurement of receiving rate r_recv

On time to send a new feedback report (t_curr - t_last > DELTA):
  calculate non-linear warping of delay d_tilde if packet loss exists
  calculate current aggregate congestion signal x_curr
  determine mode of rate adaptation for sender: rmode
  send feedback containing values of: rmode, x_curr, and r_recv
  update t_last = t_curr
```

In order for a delay-based flow to hold its ground when competing against loss-based flows (e.g., loss-based TCP), it is important to distinguish between different levels of observed queuing delay. For instance, over wired connections, a moderate queuing delay value on the order of tens of milliseonds 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.

If the last observed packet loss is within the expiration window of loss_exp (measured in terms of packet counts), the estimated queuing delay follows a non-linear warping:

$$d\_tilde = \begin{cases} d\_queue, & \text{if } d\_queue < QTH; \\ d\_queue - (QTH \cdot \exp(-LAMBDA \cdot \frac{d\_queue - QTH}{QTH})), & \text{otherwise.} \end{cases} \quad (1)$$

In (1), the queuing delay value is unchanged when it is below the first threshold QTH; otherwise it is scaled down following a non-linear curve. This non-linear warping is inspired by the delay-adaptive congestion windown backoff policy in [Budzisz-TON11], so as to "gradually nudge" the controller to operate based on loss-induced congestion signals when competing against loss-based flows. The exact form of the non-linear function has been simplified with respect to [Budzisz-TON11]. The value of the threshold QTH may need to tuned for different operational environments. Typically, a higher value of QTH is required in a noisier environment (e.g., over wireless connections, or where the video stream encounters many time-varying background competing traffic) so as to stay robust against occasional non-congestion-induced delay spikes. Additional insights on how this value can be tuned or auto-tuned should be gathered from carrying out experimental studies in different real-world deployment scenarios.

The value of loss_exp is configured to self-scale with the average packet loss interval loss_int with a multiplier MULTILOSS:

$$loss\_exp = MULTILOSS * loss\_int.$$

Estimation of the average loss interval loss_int, in turn, follows Section 5.4 of the TCP Friendly Rate Control (TFRC) protocol [RFC5348].

In practice, it is recommended to linearly interpolate between the warped (d_tilde) and non-warped (d_queue) values of the queuing delay during the transitional period lasting for the duration of loss_int.

The aggregate congestion signal is:

$$x\_curr = d\_tilde + DMARK * \left|\frac{p\_mark}{PMRREF}\right|^2 + DLOSS * \left|\frac{p\_loss}{PLRREF}\right|^2. \quad (2)$$

Here, DMARK is prescribed reference delay penalty associated with ECN markings at the reference marking ratio of PMRREF; DLOSS is prescribed reference delay penalty associated with packet losses at the reference packet loss ratio of PLRREF. The value of DLOSS and DMARK does not depend on configurations at the network node. Since ECN-enabled active queue management schemes typically mark a packet before dropping it, the value of DLOSS SHOULD be higher than that of DMARK. Furthermore, the values of DLOSS and DMARK need to be set consistently across all NADA flows sharing the same bottleneck link, so that they can compete fairly.

In the absence of packet marking and losses, the value of x_curr reduces to the observed queuing delay d_queue. In that case the NADA algorithm operates in the regime of delay-based adaptation.

Given observed per-packet delay and loss information, the receiver is also in a good position to determine whether the network is underutilized and recommend the corresponding rate adaptation mode for the sender. The criteria for operating in accelerated ramp-up mode are:

- No recent packet losses within the observation window LOGWIN; and
- No build-up of queuing delay: d_fwd-d_base < QEPS for all previous delay samples within the observation window LOGWIN.

Otherwise the algorithm operates in graduate update mode.

4.3. Sender-Side Algorithm

The sender-side algorithm is outlined as follows:

```
on initialization:
  set r_ref = RMIN
  set rtt = 0
  set x_prev = 0
  set t_last and t_curr as current system clock time

on receiving feedback report:
  obtain current timestamp from system clock: t_curr
  obtain values of rmode, x_curr, and r_recv from feedback report
  update estimation of rtt
  measure feedback interval: delta = t_curr - t_last
  if rmode == 0:
    update r_ref following accelerated ramp-up rules
  else:
    update r_ref following gradual update rules
    clip rate r_ref within the range of minimum rate (RMIN)
    and maximum rate (RMAX).
  x_prev = x_curr
  t_last = t_curr
```

In accelerated ramp-up mode, the rate r_ref is updated as follows:

$$gamma = min(GAMMA\_MAX, \frac{QBOUND}{rtt+DELTA+DFILT}) \quad (3)$$

$$r\_ref = max(r\_ref, (1+gamma)\ r\_recv) \quad (4)$$

The rate increase multiplier gamma is calculated as a function of upper bound of self-inflicted queuing delay (QBOUND), round-trip-time (rtt), target feedback interval (DELTA) and bound on filtering delay for calculating d_queue (DFILT). It has a maximum value of GAMMA_MAX. The rationale behind (3)-(4) is that the longer it takes for the sender to observe self-inflicted queuing delay build-up, the more conservative the sender should be in increasing its rate, hence the smaller the rate increase multiplier.

In gradual update mode, the rate r_ref is updated as:

$$x\_offset = x\_curr - PRIO*XREF*RMAX/r\_ref \quad (5)$$

$$x\_diff = x\_curr - x\_prev \quad (6)$$

$$r\_ref = r\_ref - KAPPA*\frac{delta}{TAU}*\frac{x\_offset}{TAU}*r\_ref - KAPPA*ETA*\frac{x\_diff}{TAU}*r\_ref \quad (7)$$

The rate changes in proportion to the previous rate decision. It is affected by two terms: offset of the aggregate congestion signal from its value at equilibrium (x_offset) and its change (x_diff). Calculation of x_offset depends on maximum rate of the flow (RMAX), its weight of priority (PRIO), as well as a reference congestion signal (XREF). The value of XREF is chosen so that the maximum rate of RMAX can be achieved when the observed congestion signal level is below PRIO*XREF.

At equilibrium, the aggregated congestion signal stablizes at x_curr = PRIO*XREF*RMAX/r_ref. This ensures that when multiple flows share the same bottleneck and observe a common value of x_curr, their rates at equilibrium will be proportional to their respective priority levels (PRIO) and the range between minimum and maximum rate. Values of the minimum rate (RMIN) and maximum rate (RMAX) will be provided by the media codec, as specified in [I-D.ietf-rmcat-cc-codec-interactions]. In the absense of such information, NADA sender will choose a default value of 0 for RMIN, and 2Mbps for RMAX.

As mentioned in the sender-side algorithm, the final rate is clipped within the dynamic range specified by the application:

$$r\_ref = \min(r\_ref, RMAX) \quad (8)$$

$$r\_ref = \max(r\_ref, RMIN) \quad (9)$$

The above operations ignore many practical issues such as clock synchronization between sender and receiver, filtering of noise in delay measurements, and base delay expiration. These will be addressed in Section 5

5. Practical Implementation of NADA

5.1. Receiver-Side Operation

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.

5.1.1. 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]. Instead of relying on RTP timestamps, the NADA sender generates its own timestamp based on local system clock and embeds that information in the transport packet header. The NADA receiver 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. By re-estimating the base delay periodically, one can avoid the potential issue of base delay expiration, whereby an earlier measured base delay value is no longer valid due to underlying route changes. All delay estimations are based on sender timestamps with a recommended granularity of 100 microseconds or higher.

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. Therefore, instead of simply calculating the value of base delay using d_base = min(d_base, d_fwd), as expressed in Section 5.1, current implementation employs a minimum filter with a window size of 15 samples over per-packet queuing delay values.

5.1.2. Estimation of packet loss/marking ratio

The receiver detects packet losses via gaps in the RTP sequence numbers of received packets. For interactive real-time media application with stringent latency constraint (e.g., video conferencing), the receiver avoids the packet re-ordering delay by treating out-of-order packets as losses. The instantaneous packet loss ratio p_inst is estimated as the ratio between the number of missing packets over the number of total transmitted packets within

the recent observation window LOGWIN. The packet loss ratio p_loss is obtained after exponential smoothing:

$$p\_loss = ALPHA*p\_inst + (1-ALPHA)*p\_loss. \quad (10)$$

The filtered result is reported back to the sender as the observed packet loss ratio p_loss.

Estimation of packet marking ratio p_mark follows the same procedure as above. It is assumed that ECN marking information at the IP header can be passed to the receiving endpoint, e.g., by following the mechanism described in [RFC6679].

5.1.3. Estimation of receiving rate

It is fairly straighforward to estimate the receiving rate r_recv. NADA maintains a recent observation window with time span of LOGWIN, and simply divides the total size of packets arriving during that window over the time span. The receiving rate (r_recv) is included as part of the feedback report.

5.2. Sender-Side Operation

Figure 4 provides a detailed view of the NADA sender. Upon receipt of an RTCP feedback report from the receiver, the NADA sender calculates the reference rate r_ref as specified in Section 4.3. It further adjusts both the target rate for the live video encoder r_vin and the sending rate r_send over the network based on the updated value of r_ref and rate shaping buffer occupancy buffer_len.

The NADA sender behavior stays the same in the presence of all types of congestion indicators: delay, loss, and ECN marking. This unified approach allows a graceful transition of the scheme as the network shifts dynamically between light and heavy congestion levels.

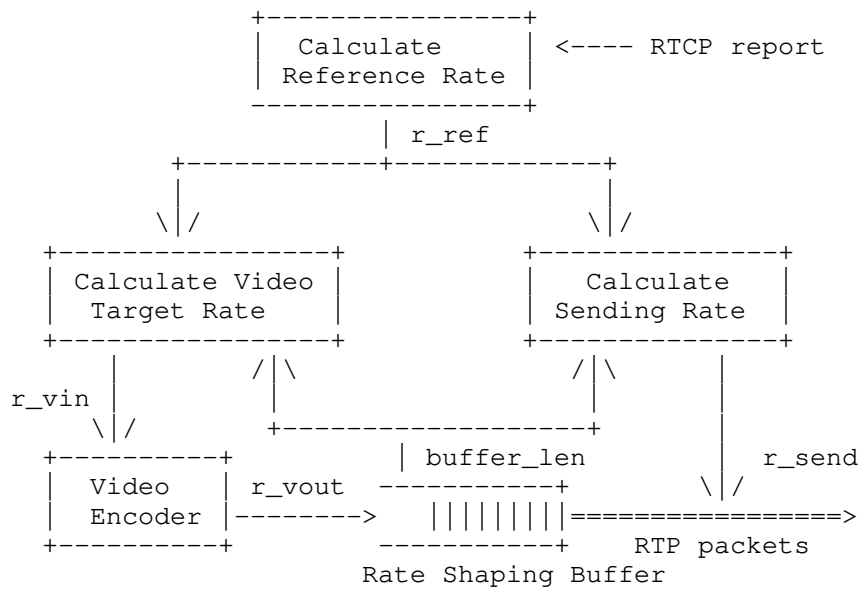


Figure 4: NADA Sender Structure

5.2.1. Rate shaping buffer

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.

A rate shaping buffer is employed to absorb any instantaneous mismatch between encoder rate output r_vout and regulated sending rate r_send. Its current level of occupancy is measured in bytes and is denoted as buffer_len.

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 prevent the rate shaping buffer from building up. The mechanisms adopted are:

- To deplete the rate shaping buffer faster by increasing the sending rate r_send; and

- To limit incoming packets of the rate shaping buffer by reducing the video encoder target rate r_vin.

5.2.2. Adjusting video target rate and sending rate

The target rate for the live video encoder deviates from the network congestion control rate r_ref based on the level of occupancy in the rate shaping buffer:

$$r_{vin} = r_{ref} - BETA_V*8*buffer_{len}*FPS. \quad (11)$$

The actual sending rate r_send is regulated in a similar fashion:

$$r_{send} = r_{ref} + BETA_S*8*buffer_{len}*FPS. \quad (12)$$

In (11) and (12), 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_ref.

Intuitively, the amount of extra rate offset needed to completely drain the rate shaping buffer within the duration of a single video frame is given by 8*buffer_len*FPS, where FPS stands for the frame rate of the video. 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 from the reference rate point.

5.3. Feedback Message Requirements

The following list of information is required for NADA congestion control to function properly:

- Recommended rate adaptation mode (rmode): a 1-bit flag indicating whether the sender should operate in accelerated ramp-up mode (rmode=0) or gradual update mode (rmode=1).

- Aggregated congestion signal (x_curr): the most recently updated value, calculated by the receiver according to Section 4.2. This information is expressed with a unit of 100 microsecond (i.e., 1/10 of a millisecond) in 15 bits. This allows a maximum value of x_curr at approximately 3.27 second.

- Receiving rate (r_recv): the most recently measured receiving rate according to Section 5.1.3. This information is expressed with a unit of bits per second (bps) in 32 bits (unsigned int). This allows a maximum rate of approximately 4.3Gbps, approximately 1000 times of the streaming rate of a typical high-definition (HD) video conferencing session today. This field can be expanded further by a few more bytes, in case an even higher rate need to be specified.

The above list of information can be accommodated by 48 bits, or 6 bytes, in total. Choice of the feedback message interval DELTA is discussed in Section 6.3 A target feedback interval of DELTA=100ms is recommended.

6. Discussions and Further Investigations

This section discussed the various design choices made by NADA, potential alternative variants of its implementation, and guidelines on how the key algorithm parameters can be chosen. Section 8 recommends additional experimental setups to further explore these topics.

6.1. Choice of 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 [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 standing queue 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 relative round-trip-time (RTT) in lieu of relative OWD, assuming that per-packet acknowledgements are available. The main drawback of RTT-based approach is the noise in the measured delay 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. Therefore, comparing the pros and cons regarding which delay metric to adopt can be kept as an orthogonal direction of investigation.

6.2. Method for delay, loss, and marking ratio estimation

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.

The current recommended practice of applying minimum filter with a window size of 15 samples suffices 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.

More sophisticated methods in packet loss ratio calculation, such as that adopted by [Floyd-CCR00], will likely be beneficial. These alternatives are currently under investigation.

6.3. Impact of parameter values

In the gradual rate update mode, the parameter TAU indicates the upper bound of round-trip-time (RTT) in feedback control loop. Typically, the observed feedback interval delta is close to the target feedback interval DELTA, and the relative ratio of delta/TAU versus ETA dictates the relative strength of influence from the aggregate congestion signal offset term (x_offset) versus its recent change (x_diff), respectively. These two terms are analogous to the integral and proportional terms in a proportional-integral (PI) controller. The recommended choice of TAU=500ms, DELTA=100ms and ETA = 2.0 corresponds to a relative ratio of 1:10 between the gains of the integral and proportional terms. Consequently, the rate adaptation is mostly driven by the change in the congestion signal with a long-term shift towards its equilibrium value driven by the offset term. Finally, the scaling parameter KAPPA determines the overall speed of the adaptation and needs to strike a balance between responsiveness and stability.

The choice of the target feedback interval DELTA needs to strike the right balance between timely feedback and low RTCP feedback message counts. A target feedback interval of DELTA=100ms is recommended, corresponding to a feedback bandwidth of 16Kbps with 200 bytes per feedback message --- approximately 1.6% overhead for a 1 Mbps flow. Furthermore, both simulation studies and frequency-domain analysis have established that a feedback interval below 250ms (i.e., more frequently than 4 feedback messages per second) will not break up the feedback control loop of NADA congestion control.

In calculating the non-linear warping of delay in (1), the current design uses fixed values of QTH for determining whether to perform the non-linear warping). Its value may need to be tuned for different operational enviornments (e.g., over wired vs. wireless connections). It is possible to adapt its value based on past observed patterns of queuing delay in the presence of packet losses. It needs to be noted that the non-linear warping mechanism may lead to multiple NADA streams stuck in loss-based mode when competing against each other.

In calculating the aggregate congestion signal x_curr, the choice of DMARK and DLOSS influence the steady-state packet loss/marking ratio experienced by the flow at a given available bandwidth. Higher values of DMARK and DLOSS result in lower steady-state loss/marking ratios, but are more susceptible to the impact of individual packet loss/marking events. While the value of DMARK and DLOSS 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 under investigation.

6.4. Sender-based vs. receiver-based calculation

In the current design, the aggregated congestion signal x_curr is calculated at the receiver, keeping the sender operation completely independent of the form of actual network congestion indications (delay, loss, or marking). Alternatively, one can move the logics of (1) and (2) to the sender. Such an approach requires slightly higher overhead in the feedback messages, which should contain individual fields on queuing delay (d_queue), packet loss ratio (p_loss), packet marking ratio (p_mark), receiving rate (r_recv), and recommended rate adaptation mode (rmode).

6.5. 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 proposed 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 [ns-2] and [ns-3] simulation platforms. Extensive ns-2 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.ietf-rmcat-eval-test] have been presented at recent IETF meetings [IETF-90][IETF-91]. Evaluation results of the current draft over several test cases in [I-D.ietf-rmcat-wireless-tests] have been presented at [IETF-93]. An open source implementation of NADA as part of a ns-3 module is available at [ns3-rmcat]

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. Suggested Experiments

NADA has been extensively evaluated under various test scenarios, including the collection of test cases specified by [I-D.ietf-rmcat-eval-test] and the subset of WiFi-based test cases in [I-D.ietf-rmcat-wireless-tests]. Additional evaluations have been carried out to characterize how NADA interacts with various active queue management (AQM) schemes such as RED, CoDel, and PIE. Most of these evaluations have been carried out in simulators. A few key test cases have been evaluated in lab environments with implementations embedded in video conferencing clients. It is strongly recommended to carry out implementation and experimentation of NADA in real-world settings. Such exercise will provide insights on how to choose to automatically adapte the values of the key algorithm parameters (see list in Figure 3) as discussed in Section 6.

Additional experiments are suggested for the following scenarios and preferrably over real-world networks:

- Experiments reflecting the set up of a typical WAN connection.
- Experiments with ECN marking capability turned on at the network for existing test cases.

- Experiments with multiple RMCAT streams bearing different user-specified priorities.

- Experiments with additional access technologies, especially over cellular networks such as 3G/LTE.

- Experiments with various media source contents, including audio only, audio and video, and application content sharing (e.g., slide shows).

9. IANA Considerations

This document makes no request of IANA.

10. Security Considerations

TBD

11. Acknowledgments

The authors would like to thank Randell Jesup, Luca De Cicco, Piers O'Hanlon, Ingemar Johansson, Stefan Holmer, Cesar Ilharco Magalhaes, Safiqul Islam, Michael Welzl, Mirja Kuhlewind, Karen Elisabeth Egede Nielsen, Julius Flohr, Roland Bless, Andreas Smas, and Martin Stiemerling for their various valuable review comments and feedback. Thanks to Charles Ganzhorn for contributing to the testbed-based evaluation of NADA during an early stage of its development.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <https://www.rfc-editor.org/info/rfc5348>.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <https://www.rfc-editor.org/info/rfc6679>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[Budzisz-TON11]
Budzisz, L., Stanojevic, R., Schlote, A., Baker, F., and R. Shorten, "On the Fair Coexistence of Loss- and Delay-Based TCP", IEEE/ACM Transactions on Networking vol. 19, no. 6, pp. 1811-1824, December 2011.

[Floyd-CCR00]
Floyd, S., Handley, M., Padhye, J., and J. Widmer, "Equation-based Congestion Control for Unicast Applications", ACM SIGCOMM Computer Communications Review vol. 30, no. 4, pp. 43-56, October 2000.

[I-D.ietf-rmcat-cc-codec-interactions]
Zanaty, M., Singh, V., Nandakumar, S., and Z. Sarker, "Congestion Control and Codec interactions in RTP Applications", draft-ietf-rmcat-cc-codec-interactions-02 (work in progress), March 2016.

[I-D.ietf-rmcat-cc-requirements]
Jesup, R. and Z. Sarker, "Congestion Control Requirements for Interactive Real-Time Media", draft-ietf-rmcat-cc-requirements-09 (work in progress), December 2014.

[I-D.ietf-rmcat-eval-test]
Sarker, Z., Singh, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-ietf-rmcat-eval-test-08 (work in progress), November 2018.

[I-D.ietf-rmcat-video-traffic-model]
Zhu, X., Cruz, S., and Z. Sarker, "Video Traffic Models for RTP Congestion Control Evaluations", draft-ietf-rmcat-video-traffic-model-06 (work in progress), November 2018.

[I-D.ietf-rmcat-wireless-tests]
Sarker, Z., Johansson, I., Zhu, X., Fu, J., Tan, W., and M. Ramalho, "Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks", draft-ietf-rmcat-wireless-tests-06 (work in progress), December 2018.

[IETF-90] Zhu, X., Ramalho, M., Ganzhorn, C., Jones, P., and R. Pan, "NADA Update: Algorithm, Implementation, and Test Case Evalua6on Results", July 2014, <https://tools.ietf.org/agenda/90/slides/slides-90-rmcat-6.pdf>.

[IETF-91] Zhu, X., Pan, R., Ramalho, M., Mena, S., Ganzhorn, C., Jones, P., and S. D'Aronco, "NADA Algorithm Update and Test Case Evaluations", November 2014, <http://www.ietf.org/proceedings/interim/2014/11/09/rmcat/slides/slides-interim-2014-rmcat-1-2.pdf>.

[IETF-93] Zhu, X., Pan, R., Ramalho, M., Mena, S., Ganzhorn, C., Jones, P., D'Aronco, S., and J. Fu, "Updates on NADA", July 2015, <https://www.ietf.org/proceedings/93/slides/slides-93-rmcat-0.pdf>.

[ns-2] "The Network Simulator - ns-2", <http://www.isi.edu/nsnam/ns/>.

[ns-3] "The Network Simulator - ns-3", <https://www.nsnam.org/>.

[ns3-rmcat]
Fu, J., Mena, S., and X. Zhu, "NS3 open source module of IETF RMCAT congestion control protocols", November 2017, <https://github.com/cisco/ns3-rmcat>.

[RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, DOI 10.17487/RFC6660, July 2012, <https://www.rfc-editor.org/info/rfc6660>.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <https://www.rfc-editor.org/info/rfc6817>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <https://www.rfc-editor.org/info/rfc7567>.

[Zhu-PV13]
Zhu, X. and R. Pan, "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.

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 variants of queue management behavior at the network node, leading to either implicit or explicit congestion signals. It needs to be acknowledged that NADA has not yet been tested with non-probabilistic ECN marking behaviors.

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. The system can scale easily with a large number of video flows and at high link capacity.

A.1. Default behavior of drop tail queues

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 aggregated congestion signal x_curr. No special action is required at network node.

A.2. RED-based 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 [RFC7567]. Calculation of the marking probability involves the following steps:

on packet arrival:
update smoothed queue size q_avg as:
q_avg = w*q + (1-w)*q_avg.

calculate marking probability p as:

$$p = \begin{cases} 0, & \text{if } q < q\_lo; \\ p\_max * \dfrac{q\_avg - q\_lo}{q\_hi - q\_lo}, & \text{if } q\_lo <= q < q\_hi; \\ p = 1, & \text{if } q >= q\_hi. \end{cases}$$

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_curr at the receiver. No changes are required at the sender.

A.3. Random Early Marking with Virtual Queues

Advanced network nodes may support random early marking based on a token bucket algorithm originally designed for Pre-Congestion Notification (PCN) [RFC6660]. The early congestion notification (ECN) bit in the IP header of packets are marked randomly. The marking probability is calculated based on a token-bucket algorithm originally designed for the Pre-Congestion Notification (PCN) [RFC6660]. The target link utilization is set as 90%; the marking probability is designed to grow linearly with the token bucket size when it varies between 1/3 and 2/3 of the full token bucket limit.

Calculation of the marking probability involves the following steps:

```
    upon packet arrival:
        meter packet against token bucket (r,b);

        update token level b_tk;

        calculate the marking probability as:

         / 0,                      if b-b_tk < b_lo;
         |
         |          b-b_tk-b_lo
    p = <  p_max* --------------, if b_lo<= b-b_tk <b_hi;
         |           b_hi-b_lo
         |
         \ 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 to be below capacity, resulting in slight under-utilization of the link. The maximum marking probability is p_max.

The ECN markings events will contribute to the calculation of an equivalent delay x_curr at the receiver. No changes are required at the sender. The virtual queuing mechanism from the PCN-based marking algorithm will lead to additional benefits such as zero standing queues.

Authors' Addresses

Xiaoqing Zhu
Cisco Systems
12515 Research Blvd., Building 4
Austin, TX 78759
USA

Email: xiaoqzhu@cisco.com

Rong Pan *
* Pending affiliation change.

Email: rong.pan@gmail.com

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

Phone: +1 919 476 2038
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

Paul E. Jones
Cisco Systems
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Email: paulej@packetizer.com

Jiantao Fu
Cisco Systems
707 Tasman Drive
Milpitas, CA 95035
USA

Email: jianfu@cisco.com

Stefano D'Aronco
Ecole Polytechnique Federale de Lausanne
EPFL STI IEL LTS4, ELD 220 (Batiment ELD), Station 11
Lausanne CH-1015
Switzerland

Email: stefano.daronco@epfl.ch

RTP Media Congestion Avoidance Techniques
Internet-Draft
Intended status: Experimental
Expires: September 30, 2018

D. Hayes, Ed.
Simula Research Laboratory
S. Ferlin

M. Welzl
K. Hiorth
University of Oslo

March 29, 2018

Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.
draft-ietf-rmcat-sbd-11

Abstract

This document describes a mechanism to detect whether end-to-end data flows share a common bottleneck. It relies on summary statistics that are calculated based on continuous measurements and used as input to a grouping algorithm that runs wherever the knowledge is needed.

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 https://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 30, 2018.

Copyright Notice

Table of Contents

1. Introduction

In the Internet, it is not normally known if flows (e.g., TCP connections or UDP data streams) traverse the same bottlenecks. Even flows that have the same sender and receiver may take different paths and may or may not share a bottleneck. Flows that share a bottleneck link usually compete with one another for their share of the capacity. This competition has the potential to increase packet loss and delays. This is especially relevant for interactive applications that communicate simultaneously with multiple peers (such as multi-party video). For RTP media applications such as RTCWEB, [I-D.ietf-rmcat-coupled-cc] describes a scheme that combines the congestion controllers of flows in order to honor their priorities and avoid unnecessary packet loss as well as delay. This mechanism relies on some form of Shared Bottleneck Detection (SBD); here, a measurement-based SBD approach is described.

1.1. The Basic Mechanism

The mechanism groups flows that have similar statistical characteristics together. Section 3.3.1 describes a simple method for achieving this, however, a major part of this draft is concerned with collecting suitable statistics for this purpose.

1.2. The Signals

The current Internet is unable to explicitly inform endpoints as to which flows share bottlenecks, so endpoints need to infer this from whatever information is available to them. The mechanism described here currently utilizes packet loss and packet delay, but is not restricted to these. As ECN becomes more prevalent it too will become a valuable base signal.

1.2.1. Packet Loss

Packet loss is often a relatively infrequent indication that a flow traverses a bottleneck. Therefore, on its own it is of limited use for SBD, however, it is a valuable supplementary measure when it is more prevalent (refer to [RFC2680] section 2.5 for measuring packet loss).

1.2.2. Packet Delay

End-to-end delay measurements include noise from every device along the path in addition to the delay perturbation at the bottleneck device. The noise is often significantly increased if the round-trip time is used. The cleanest signal is obtained by using One-Way-Delay (OWD) (refer to [RFC7679] section 3 for a definition of OWD).

Measuring absolute OWD is difficult since it requires both the sender and receiver clocks to be synchronized. However, since the statistics being collected are relative to the mean OWD, a relative OWD measurement is sufficient. Clock skew is not usually significant over the time intervals used by this SBD mechanism (see [RFC6817] A.2 for a discussion on clock skew and OWD measurements). However, in circumstances where it is significant, Section 5.2 outlines a way of adjusting the calculations to cater for it.

Each packet arriving at the bottleneck buffer may experience very different queue lengths, and therefore different waiting times. A single OWD sample does not, therefore, characterize the path well. However, multiple OWD measurements do reflect the distribution of delays experienced at the bottleneck.

1.2.3. Path Lag

Flows that share a common bottleneck may traverse different paths, and these paths will often have different base delays. This makes it difficult to correlate changes in delay or loss. This technique uses the long term shape of the delay distribution as a base for comparison to counter this.

2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] RFC2119 [RFC2119] RFC8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

Acronyms used in this document:

OWD -- One Way Delay

MAD -- Mean Absolute Deviation

RTT -- Round Trip Time

SBD -- Shared Bottleneck Detection

Conventions used in this document:

T	--	the base time interval over which measurements are made
N	--	the number of base time, T, intervals used in some calculations
M	--	the number of base time, T, intervals used in some calculations, where M <= N
sum(...)	--	summation of terms of the variable in parentheses
sum_T(...)	--	summation of all the measurements of the variable in parentheses taken over the interval T
sum_NT(...)	--	summation of all measurements taken over the interval N*T
sum_MT(...)	--	summation of all measurements taken over the interval M*T
E_T(...)	--	the expectation or mean of the measurements of the variable in parentheses over T
E_N(...)	--	the expectation or mean of the last N values of the variable in parentheses
E_M(...)	--	the expectation or mean of the last M values of the variable in parentheses
num_T(...)	--	the count of measurements of the variable in parentheses taken in the interval T
num_MT(...)	--	the count of measurements of the variable in parentheses taken in the interval NT
PB	--	a boolean variable indicating the particular flow was identified transiting a bottleneck in the previous interval T (i.e. Previously Bottleneck)
skew_est	--	a measure of skewness in a OWD distribution
skew_base_T	--	a variable used as an intermediate step in calculating skew_est
var_est	--	a measure of variability in OWD measurements

var_base_T -- a variable used as an intermediate step in calculating var_est

freq_est -- a measure of low frequency oscillation in the OWD measurements

pkt_loss -- a measure of the proportion of packets lost

p_l, p_f, p_mad, c_s, c_h, p_s, p_d, p_v -- various thresholds used in the mechanism

M and F -- number of values related to N

2.1. Parameters and Their Effect

T T should be long enough so that there are enough packets received during T for a useful estimate of short term mean OWD and variation statistics. Making T too large can limit the efficacy of freq_est. It will also increase the response time of the mechanism. Making T too small will make the metrics noisier.

N & M N should be large enough to provide a stable estimate of oscillations in OWD. Usually M=N, though having M<N may be beneficial in certain circumstances. M*T needs to be long enough to provide stable estimates of skewness and MAD.

F F determines the number of intervals over which statistics are considered to be equally weighted. When F=M recent and older measurements are considered equal. Making F<M can increase the responsiveness of the SBD mechanism. If F is too small, statistics will be too noisy.

c_s c_s is the threshold in skew_est used for determining whether a flow is transiting a bottleneck or not. Lower values of c_s require bottlenecks to be more congested to be considered for grouping by the mechanism. c_s should be set within the range of +0.2 to -0.1; low enough so that lightly loaded paths do not give a false indication.

p_l p_l is the threshold in pkt_loss used for determining whether a flow is transiting a bottleneck or not. When pkt_loss is high it becomes a better indicator of congestion than skew_est.

c_h c_h adds hysteresis to the bottleneck determination. It should be large enough to avoid constant switching in the determination, but low enough to ensure that grouping is not attempted when there is no bottleneck and the delay and loss signals cannot be relied upon.

p_v p_v determines the sensitivity of freq_est to noise. Making it smaller will yield higher but noisier values for freq_est. Making it too large will render it ineffective for determining groups.

p_* Flows are separated when the skew_est|var_est|freq_est|pkt_loss measure is greater than p_s|p_mad|p_f|p_d. Adjusting these is a compromise between false grouping of flows that do not share a bottleneck and false splitting of flows that do. Making them larger can help if the measures are very noisy, but reducing the noise in the statistical measures by adjusting T and N|M may be a better solution.

2.2. Recommended Parameter Values

Reference [Hayes-LCN14] uses T=350ms, N=50, p_l=0.1. The other parameters have been tightened to reflect minor enhancements to the algorithm outlined in Section 4: c_s=0.1, p_f=p_d=0.1, p_s=0.15, p_mad=0.1, p_v=0.7. M=30, F=20, and c_h = 0.3 are additional parameters defined in the document. These are values that seem to work well over a wide range of practical Internet conditions.

3. Mechanism

The mechanism described in this document is based on the observation that the distribution of delay measurements of packets that traverse a common bottleneck have similar shape characteristics. These shape characteristics are described using 3 key summary statistics:

variability (estimate var_est, see Section 3.2.3)

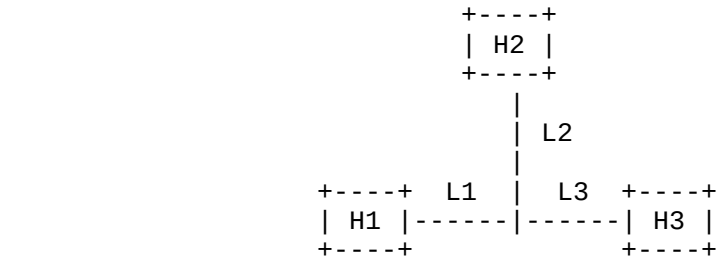
skewness (estimate skew_est, see Section 3.2.2)

oscillation (estimate freq_est, see Section 3.2.4)

with packet loss (estimate pkt_loss, see Section 3.2.5) used as a supplementary statistic.

Summary statistics help to address both the noise and the path lag problems by describing the general shape over a relatively long period of time. Each summary statistic portrays a "view" of the

bottleneck link characteristics, and when used together, they provide a robust discrimination for grouping flows. An RTP Media device may be both a sender and a receiver and SBD can be performed at either a sender or a receiver or both.



A network with 3 hosts (H1, H2, H3) and 3 links (L1, L2, L3).

Figure 1

In Figure 1, there are two possible locations for shared bottleneck detection: sender-side and receiver-side.

1. Sender-side: consider a situation where host H1 sends media streams to hosts H2 and H3, and L1 is a shared bottleneck. H2 and H3 measure the OWD and packet loss and either send back this raw data, or the calculated summary statistics, periodically to H1 every T. H1, having this knowledge, can determine the shared bottleneck and accordingly control the send rates.

2. Receiver-side: consider that H2 is also sending media to H3, and L3 is a shared bottleneck. If H3 sends summary statistics to H1 and H2, neither H1 nor H2 alone obtain enough knowledge to detect this shared bottleneck; H3 can however determine it by combining the summary statistics related to H1 and H2, respectively.

3.1. SBD Feedback Requirements

There are three possible scenarios each with different feedback requirements:

1. Both summary statistic calculations and SBD are performed at senders only. When sender-based congestion control is implemented, this method is RECOMMENDED.

2. Summary statistics calculated on the receivers and SBD at the senders.

3. Summary statistic calculations on receivers, and SBD performed at both senders and receivers (beyond the current scope, but allows cooperative detection of bottlenecks).

All three possibilities are discussed for completeness in this document, however, it is expected that feedback will take the form of scenario 1 and operate in conjunction with sender-based congestion control mechanisms.

3.1.1. Feedback When All the Logic is Placed at the Sender

Having the sender calculate the summary statistics and determine the shared bottlenecks based on them has the advantage of placing most of the functionality in one place -- the sender.

For every packet, the sender requires accurate relative OWD measurements of adequate precision, along with an indication of lost packets (or the proportion of packets lost over an interval). An method to provide such measurement data with RTCP is described in [I-D.ietf-avtcore-cc-feedback-message].

Sums, var_base_T and skew_base_T are calculated incrementally as relative OWD measurements are determined from the feedback messages. When the mechanism has received sufficient measurements to cover the T base time interval for all flows, the summary statistics (see Section 3.2) are calculated for that T interval and flows are grouped (see Section 3.3.1). The exact timing of these calculations will depend on the frequency of the feedback message.

3.1.2. Feedback When the Statistics are Calculated at the Receiver and SBD Performed at the Sender

This scenario minimizes feedback, but requires receivers to send selected summary statistics at an agreed regular interval. We envisage the following exchange of information to initialize the system:

- An initialization message from the sender to the receiver will contain the following information:
  - A list of which key metrics should be collected and relayed back to the sender out of a possibly extensible set (pkt_loss, var_est, skew_est, freq_est). The grouping algorithm described in this document requires all four of these metrics, and receivers MUST be able to provide them, but future algorithms may be able to exploit other metrics (e.g. metrics based on explicit network signals).

    * The values of T, N, M, and the necessary resolution and precision of the relayed statistics.

  o A response message from the receiver acknowledges this message with a list of key metrics it supports (subset of the senders list) and is able to relay back to the sender.

This initialization exchange may be repeated to finalize the agreed metrics should not all be supported by all receivers. It is also recommendable to include an identifier for the SBD algorithm version in the initialization message from the sender, so that potential advances in SBD technology can be easily deployed. For reference, the mechanism outlined in this document has the identifier SBD=01.

After initialization the agreed summary statistics are fed back to the sender (nominally every T).

3.1.3. Feedback When Bottlenecks can be Determined at Both Senders and Receivers

This type of mechanism is currently beyond the scope of the SBD algorithm described in this document. It is mentioned here to ensure more advanced sender/receiver cooperative shared bottleneck determination mechanisms remain possible in the future.

It is envisaged that such a mechanism would be initialized in a similar manner to that described in Section 3.1.2.

After initialization both summary statistics and shared bottleneck determinations should be exchanged, nominally every T.

3.2. Key Metrics and Their Calculation

Measurements are calculated over a base interval, T and summarized over N or M such intervals. All summary statistics can be calculated incrementally.

3.2.1. Mean Delay

The mean delay is not a useful signal for comparisons between flows since flows may traverse quite different paths and clocks will not necessarily be synchronized. However, it is a base measure for the 3 summary statistics. The mean delay, E_T(OWD), is the average one way delay measured over T.

To facilitate the other calculations, the last N E_T(OWD) values will need to be stored in a cyclic buffer along with the moving average of E_T(OWD):

mean_delay = E_M(E_T(OWD)) = sum_M(E_T(OWD)) / M

where M <= N. Setting M to be less than N allows the mechanism to be more responsive to changes, but potentially at the expense of a higher error rate (see Section 4.1 for a discussion on improving the responsiveness of the mechanism.)

3.2.2. Skewness Estimate

Skewness is difficult to calculate efficiently and accurately. Ideally it should be calculated over the entire period (M * T) from the mean OWD over that period. However this would require storing every delay measurement over the period. Instead, an estimate is made over M * T based on a calculation every T using the previous T's calculation of mean_delay.

The base for the skewness calculation is estimated using a counter initialized every T. It increments for one way delay (OWD) samples below the mean and decrements for OWD above the mean. So for each OWD sample:

if (OWD < mean_delay) skew_base_T++

if (OWD > mean_delay) skew_base_T--

The mean_delay does not include the mean of the current T interval to enable it to be calculated iteratively.

skew_est = sum_MT(skew_base_T)/num_MT(OWD)

where skew_est is a number between -1 and 1

Note: Care must be taken when implementing the comparisons to ensure that rounding does not bias skew_est. It is important that the mean is calculated with a higher precision than the samples.

3.2.3. Variability Estimate

Mean Absolute Deviation (MAD) delay is a robust variability measure that copes well with different send rates. It can be implemented in an online manner as follows:

```
var_base_T = sum_T(|OWD - E_T(OWD)|)

   where

      |x| is the absolute value of x

      E_T(OWD) is the mean OWD calculated in the previous T

var_est = MAD_MT = sum_MT(var_base_T)/num_MT(OWD)
```

3.2.4. Oscillation Estimate

An estimate of the low frequency oscillation of the delay signal is calculated by counting and normalizing the significant mean, E_T(OWD), crossings of mean_delay:

```
freq_est = number_of_crossings / N

   where we define a significant mean crossing as a crossing that
   extends p_v * var_est from mean_delay. In our experiments we
   have found that p_v = 0.7 is a good value.
```

Freq_est is a number between 0 and 1. Freq_est can be approximated incrementally as follows:

With each new calculation of E_T(OWD) a decision is made as to whether this value of E_T(OWD) significantly crosses the current long term mean, mean_delay, with respect to the previous significant mean crossing.

A cyclic buffer, last_N_crossings, records a 1 if there is a significant mean crossing, otherwise a 0.

The counter, number_of_crossings, is incremented when there is a significant mean crossing and decremented when a non-zero value is removed from the last_N_crossings.

This approximation of freq_est was not used in [Hayes-LCN14], which calculated freq_est every T using the current E_N(E_T(OWD)). Our tests show that this approximation of freq_est yields results that are almost identical to when the full calculation is performed every T.

3.2.5. Packet Loss

The proportion of packets lost over the period NT is used as a supplementary measure:

```
pkt_loss = sum_NT(lost packets) / sum_NT(total packets)
```

Note: When pkt_loss is small it is very variable, however, when pkt_loss is high it becomes a stable measure for making grouping decisions.

3.3. Flow Grouping

3.3.1. Flow Grouping Algorithm

The following grouping algorithm is RECOMMENDED for use of SBD with coupled congestion control for RTP media [I-D.ietf-rmcat-coupled-cc] and is sufficient and efficient for small to moderate numbers of flows. For very large numbers of flows (e.g. hundreds), a more complex clustering algorithm may be substituted.

Since no single metric is precise enough to group flows (due to noise), the algorithm uses multiple metrics. Each metric offers a different "view" of the bottleneck link characteristics, and used together they enable a more precise grouping of flows than would otherwise be possible.

Flows determined to be transiting a bottleneck are successively divided into groups based on freq_est, var_est, skew_est and pkt_loss.

The first step is to determine which flows are transiting a bottleneck. This is important, since if a flow is not transiting a bottleneck its delay based metrics will not describe the bottleneck, but the "noise" from the rest of the path. Skewness, with proportion of packet loss as a supplementary measure, is used to do this:

1. Grouping will be performed on flows that are inferred to be traversing a bottleneck by:

   skew_est < c_s

      || (skew_est < c_h & PB) || pkt_loss > p_l

   The parameter c_s controls how sensitive the mechanism is in detecting a bottleneck. c_s = 0.0 was used in [Hayes-LCN14]. A value of c_s = 0.1 is a little more sensitive, and c_s = -0.1 is a little less sensitive. c_h controls the hysteresis on flows that were grouped as transiting a bottleneck last time. If the test result is TRUE, PB=TRUE, otherwise PB=FALSE.

   These flows, flows transiting a bottleneck, are then progressively divided into groups based on the freq_est, var_est, and skew_est summary statistics. The process proceeds according to the following steps:

2. Group flows whose difference in sorted freq_est is less than a threshold:

   diff(freq_est) < p_f

3. Subdivide the groups obtained in 2. by grouping flows whose difference in sorted E_M(var_est) (highest to lowest) is less than a threshold:

   diff(var_est) < (p_mad * var_est)

   The threshold, (p_mad * var_est), is with respect to the highest value in the difference.

4. Subdivide the groups obtained in 3. by grouping flows whose difference in sorted skew_est is less than a threshold:

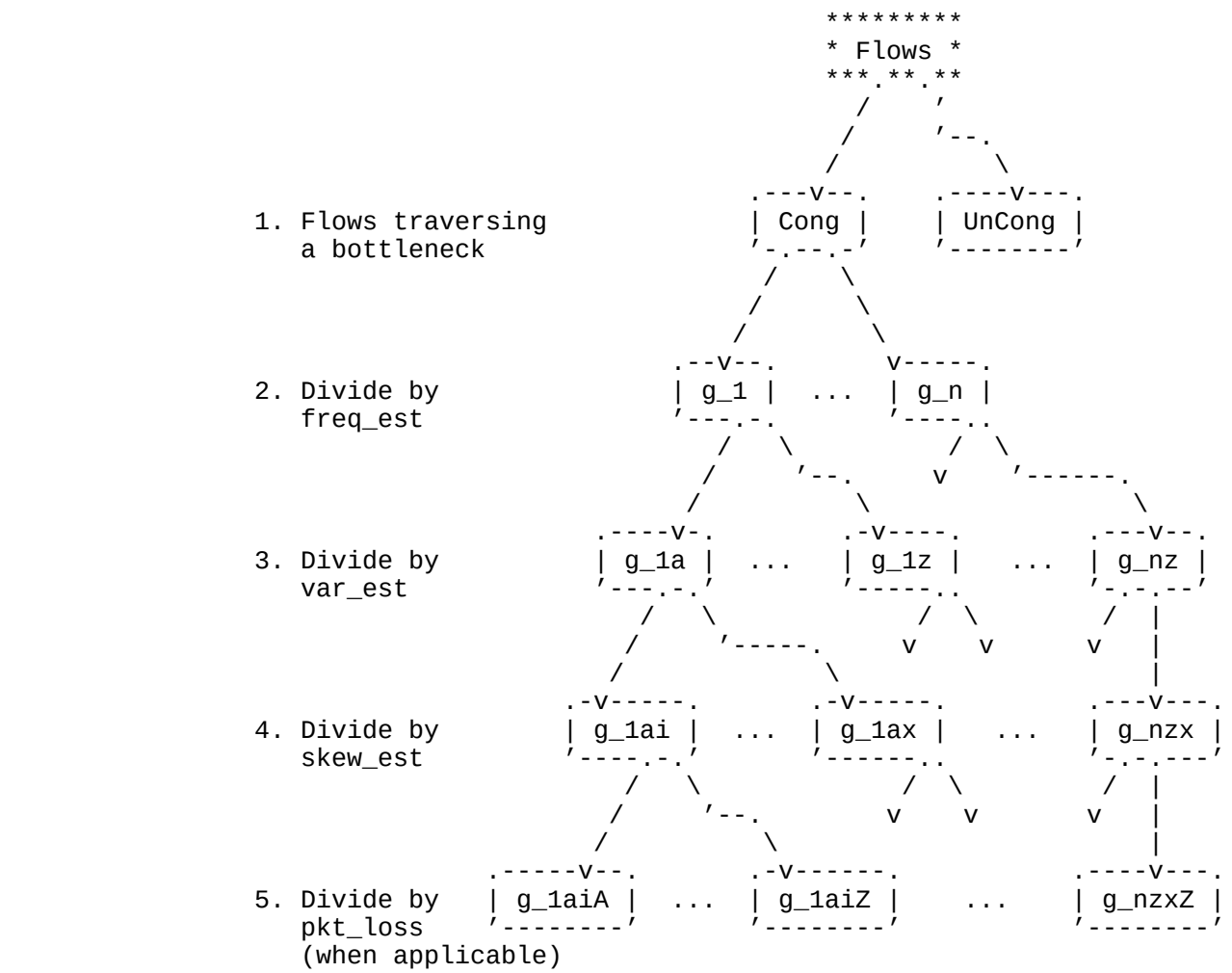
   diff(skew_est) < p_s

5. When packet loss is high enough to be reliable (pkt_loss > p_l), Subdivide the groups obtained in 4. by grouping flows whose difference is less than a threshold

   diff(pkt_loss) < (p_d * pkt_loss)

   The threshold, (p_d * pkt_loss), is with respect to the highest value in the difference.

This procedure involves sorting estimates from highest to lowest. It is simple to implement, and efficient for small numbers of flows (up to 10-20). Figure 2 illustrates this algorithm.



Simple grouping algorithm.

Figure 2

3.3.2. Using the Flow Group Signal

Grouping decisions can be made every T from the second T, however they will not attain their full design accuracy until after the 2*N'th T interval. We recommend that grouping decisions are not made until 2*M T intervals.

Network conditions, and even the congestion controllers, can cause bottlenecks to fluctuate. A coupled congestion controller MAY decide only to couple groups that remain stable, say grouped together 90% of the time, depending on its objectives. Recommendations concerning this are beyond the scope of this document and will be specific to the coupled congestion controller's objectives.

4. Enhancements to the Basic SBD Algorithm

The SBD algorithm as specified in Section 3 was found to work well for a broad variety of conditions. The following enhancements to the basic mechanisms have been found to significantly improve the algorithm's performance under some circumstances and SHOULD be implemented. These "tweaks" are described separately to keep the main description succinct.

4.1. Reducing Lag and Improving Responsiveness

This section describes how to improve the responsiveness of the basic algorithm.

Measurement based shared bottleneck detection makes decisions in the present based on what has been measured in the past. This means that there is always a lag in responding to changing conditions. This mechanism is based on summary statistics taken over (N*T) seconds. This mechanism can be made more responsive to changing conditions by:

1. Reducing N and/or M -- but at the expense of having less accurate metrics, and/or

2. Exploiting the fact that more recent measurements are more valuable than older measurements and weighting them accordingly.

Although more recent measurements are more valuable, older measurements are still needed to gain an accurate estimate of the distribution descriptor we are measuring. Unfortunately, the simple exponentially weighted moving average weights drop off too quickly for our requirements and have an infinite tail. A simple linearly declining weighted moving average also does not provide enough weight to the most recent measurements. We propose a piecewise linear distribution of weights, such that the first section (samples 1:F) is

flat as in a simple moving average, and the second section (samples F+1:M) is linearly declining weights to the end of the averaging window. We choose integer weights, which allows incremental calculation without introducing rounding errors.

4.1.1. Improving the Response of the Skewness Estimate

The weighted moving average for skew_est, based on skew_est in Section 3.2.2, can be calculated as follows:

```
skew_est = ((M-F+1)*sum(skew_base_T(1:F))
                + sum([(M-F):1].*skew_base_T(F+1:M)))
           / ((M-F+1)*sum(numsampT(1:F))
                + sum([(M-F):1].*numsampT(F+1:M)))
```

where numsampT is an array of the number of OWD samples in each T (i.e. num_T(OWD)), and numsampT(1) is the most recent; skew_base_T(1) is the most recent calculation of skew_base_T; 1:F refers to the integer values 1 through to F, and [(M-F):1] refers to an array of the integer values (M-F) declining through to 1; and ".*" is the array scalar dot product operator.

To calculate this weighted skew_est incrementally:

Notation: F_ - flat portion, D_ - declining portion, W_ - weighted component

Initialize: sum_skewbase = 0, F_skewbase=0, W_D_skewbase=0

skewbase_hist = buffer length M initialize to 0

numsampT = buffer length M initialized to 0

Steps per iteration:

1. old_skewbase = skewbase_hist(M)

2. old_numsampT = numsampT(M)

3. cycle(skewbase_hist)

4. cycle(numsampT)

5. numsampT(1) = num_T(OWD)

6. skewbase_hist(1) = skew_base_T

7. F_skewbase = F_skewbase + skew_base_T - skewbase_hist(F+1)

8. W_D_skewbase = W_D_skewbase + (M-F)*skewbase_hist(F+1) - sum_skewbase

9. W_D_numsamp = W_D_numsamp + (M-F)*numsampT(F+1) - sum_numsamp + F_numsamp

10. F_numsamp = F_numsamp + numsampT(1) - numsampT(F+1)

11. sum_skewbase = sum_skewbase + skewbase_hist(F+1) - old_skewbase

12. sum_numsamp = sum_numsamp + numsampT(1) - old_numsampT

13. skew_est = ((M-F+1)*F_skewbase + W_D_skewbase) / ((M-F+1)*F_numsamp+W_D_numsamp)

Where cycle(....) refers to the operation on a cyclic buffer where the start of the buffer is now the next element in the buffer.

4.1.2. Improving the Response of the Variability Estimate

Similarly the weighted moving average for var_est can be calculated as follows:

```
var_est =  ((M-F+1)*sum(var_base_T(1:F))
                + sum([(M-F):1].*var_base_T(F+1:M)))
           / ((M-F+1)*sum(numsampT(1:F))
                + sum([(M-F):1].*numsampT(F+1:M)))
```

where numsampT is an array of the number of OWD samples in each T (i.e. num_T(OWD)), and numsampT(1) is the most recent; skew_base_T(1) is the most recent calculation of skew_base_T; 1:F refers to the integer values 1 through to F, and [(M-F):1] refers to an array of the integer values (M-F) declining through to 1; and ".*" is the array scalar dot product operator. When removing oscillation noise (see Section 4.2) this calculation must be adjusted to allow for invalid var_base_T records.

Var_est can be calculated incrementally in the same way as skew_est in Section 4.1.1. However, note that the buffer numsampT is used for both calculations so the operations on it should not be repeated.

4.2. Removing Oscillation Noise

When a path has no bottleneck, var_est will be very small and the recorded significant mean crossings will be the result of path noise. Thus up to N-1 meaningless mean crossings can be a source of error at the point a link becomes a bottleneck and flows traversing it begin to be grouped.

To remove this source of noise from freq_est:

1. Set the current var_base_T = NaN (a value representing an invalid record, i.e. Not a Number) for flows that are deemed to not be transiting a bottleneck by the first skew_est based grouping test (see Section 3.3.1).

2. Then var_est = sum_MT(var_base_T != NaN) / num_MT(OWD)

3. For freq_est, only record a significant mean crossing if flow deemed to be transiting a bottleneck.

These three changes can help to remove the non-bottleneck noise from freq_est.

5. Measuring OWD

This section discusses the OWD measurements required for this algorithm to detect shared bottlenecks.

The SBD mechanism described in this document relies on differences between OWD measurements to avoid the practical problems with measuring absolute OWD (see [Hayes-LCN14] section IIIC). Since all summary statistics are relative to the mean OWD and sender/receiver clock offsets should be approximately constant over the measurement periods, the offset is subtracted out in the calculation.

5.1. Time-stamp Resolution

The SBD mechanism requires timing information precise enough to be able to make comparisons. As a rule of thumb, the time resolution should be less than one hundredth of a typical path's range of delays. In general, the coarser the time resolution, the more care that needs to be taken to ensure rounding errors do not bias the skewness calculation. Frequent timing information in millisecond resolution as described by [I-D.ietf-avtcore-cc-feedback-message] should be sufficient for the sender to calculate relative OWD.

5.2. Clock Skew

Generally sender and receiver clock skew will be too small to cause significant errors in the estimators. Skew_est and freq_est are the most sensitive to this type of noise due to their use of a mean OWD calculated over a longer interval. In circumstances where clock skew is high, basing skew_est only on the previous T's mean and ignoring freq_est provides a noisier but reliable signal.

A more sophisticated method is to estimate the effect the clock skew is having on the summary statistics, and then adjust statistics accordingly. There are a number of techniques in the literature, including [Zhang-Infocom02].

6. Expected Feedback from Experiments

The algorithm described in this memo has so far been evaluated using simulations and small scale experiments. Real network tests using RTP Media Congestion Avoidance Techniques (RMCAT) congestion control algorithms will help confirm the default parameter choice. For example, the time interval T may need to be made longer if the packet

rate is very low. Implementers and testers are invited to document their findings in an Internet draft.

7. Acknowledgments

This work was part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the authors.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

The security considerations of RFC 3550 [RFC3550], RFC 4585 [RFC4585], and RFC 5124 [RFC5124] are expected to apply.

Non-authenticated RTCP packets carrying OWD measurements, shared bottleneck indications, and/or summary statistics could allow attackers to alter the bottleneck sharing characteristics for private gain or disruption of other parties' communication. When using SBD for coupled congestion control as described in [I-D.ietf-rmcat-coupled-cc], the security considerations of [I-D.ietf-rmcat-coupled-cc] apply.

10. Change history

XX RFC ED - PLEASE REMOVE THIS SECTION XXX

Changes made to this document:

WG-10->WG-11 : Genart review addressed.

WG-09->WG-10 : AD review addressed.

WG-08->WG-09 : Removed definitions that are no longer used. Added pkt_loss definition. Refined c_s recommendation.

WG-07->WG-08 : Updates addressing https://www.ietf.org/mail-archive/web/rmcat/current/msg01671.html Mainly clarifications.

WG-06->WG-07 : Updates addressing https://mailarchive.ietf.org/arch/msg/rmcat/80B6q4nI7carGcf_ddBwx7nKvOw. Mainly

clarifications. Figure 2 to supplement grouping algorithm description.

WG-05->WG-06 : Updates addressing WG reviews https://mailarchive.ietf.org/arch/msg/rmcat/-1JdrTMq1Y5T6ZNlOkrQJQ27TzE and https://mailarchive.ietf.org/arch/msg/rmcat/eI2Q1f8NL2SxbJgjFLR4_rEmJ_g. This has mainly involved minor clarifications, including the moving of 3.4.1 and 3.5 into the new Section 4, and 3.4.1 into Section 5

WG-04->WG-05 : Fix ToC formatting. Add section on expected feedback from experiments replacing short section on implementation status. Added comment on ECN as a signal. Clarification of lost packet signaling. Change term "draft" to "document" where appropriate. American spelling. Some tightening of the text.

WG-03->WG-04 : Add M to terminology table, suggest skew_est based on previous T and no freq_est in clock skew section, feedback requirements as a separate sub section.

WG-02->WG-03 : Correct misspelled author

WG-01->WG-02 : Removed ambiguity associated with the term "congestion". Expanded the description of initialization messages. Removed PDV metric. Added description of incremental weighted metric calculations for skew_est. Various clarifications based on implementation work. Fixed typos and tuned parameters.

WG-00->WG-01 : Moved unbiased skew section to replace skew estimate, more robust variability estimator, the term variance replaced with variability, clock drift term corrected to clock skew, revision to clock skew section with a place holder, description of parameters.

02->WG-00 : Fixed missing 0.5 in 3.3.2 and missing brace in 3.3.3

01->02 : New section describing improvements to the key metric calculations that help to remove noise, bias, and reduce lag. Some revisions to the

notation to make it clearer. Some tightening of the thresholds.

00->01 : Revisions to terminology for clarity

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

11.2. Informative References

[Hayes-LCN14] Hayes, D., Ferlin, S., and M. Welzl, "Practical Passive Shared Bottleneck Detection using Shape Summary Statistics", Proc. the IEEE Local Computer Networks (LCN) pp150-158, September 2014, <http://heim.ifi.uio.no/davihay/hayes14__pract_passiv_shared_bottl_detec-abstract.html>.

[I-D.ietf-avtcore-cc-feedback-message] Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", draft-ietf-avtcore-cc-feedback-message-01 (work in progress), March 2018.

[I-D.ietf-rmcat-coupled-cc] Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-07 (work in progress), September 2017.

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, DOI 10.17487/RFC2680, September 1999, <https://www.rfc-editor.org/info/rfc2680>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[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, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/rfc4585>.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <https://www.rfc-editor.org/info/rfc5124>.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <https://www.rfc-editor.org/info/rfc6817>.

[RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <https://www.rfc-editor.org/info/rfc7679>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[Zhang-Infocom02] Zhang, L., Liu, Z., and H. Xia, "Clock synchronization algorithms for network measurements", Proc. the IEEE International Conference on Computer Communications (INFOCOM) pp160-169, September 2002, <http://dx.doi.org/10.1109/INFCOM.2002.1019257>.

Authors' Addresses

David Hayes (editor)
Simula Research Laboratory
P.O. Box 134
Lysaker 1325
Norway

Email: davidh@simula.no

Simone Ferlin

Email: simone@ferlin.io

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

Email: michawe@ifi.uio.no

Kristian Hiorth
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Email: kristahi@ifi.uio.no

RTP Media Congestion Avoidance Techniques (rmcat)
Internet-Draft
Intended status: Experimental
Expires: December 20, 2015

M. Welzl
S. Islam
S. Gjessing
University of Oslo

June 18, 2015

Coupled congestion control for RTP media
draft-welzl-rmcat-coupled-cc-05

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. It specifies how to apply the method for the NADA congestion control algorithm.

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 20, 2015.

Copyright Notice

Table of Contents

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.

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 often only available at the sender side, making the sender an obvious choice for placement of the elements and mechanisms described here.

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).

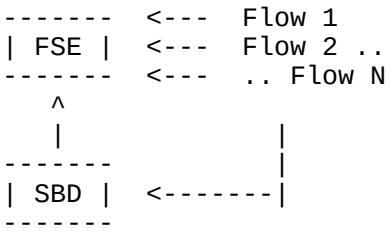


Figure 1: Coupled congestion control architecture

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 in three basic ways:

1. From multiplexing: 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].

2. Via configuration: e.g. by assuming that a common wireless uplink is also a shared bottleneck.

3. From measurements: e.g. by considering correlations among measured delay and loss as an indication of a shared bottleneck.

The methods above have some essential trade-offs: e.g., multiplexing is a completely reliable measure, however it is limited in scope to two end points (i.e., it cannot be applied to couple congestion controllers of one sender talking to multiple receivers). A measurement-based SBD mechanism is described in [sbd]. Measurements can never be 100% reliable, in particular because they are based on the past but applying coupled congestion control means to make an assumption about the future; it is therefore recommended to implement cautionary measures, e.g. by disabling coupled congestion control if enabling it causes a significant increase in delay and/or packet loss. Measurements also take time, which entails a certain delay for turning on coupling (refer to [sbd] for details).

5.2. FSE

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

- a unique flow number to identify the flow
- 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)
- 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
- The rate used by the flow in bits per second, FSE_R.

The FSE can operate on window-based as well as rate-based congestion controllers (TEMPORARY NOTE: and probably -- not yet tested -- combinations thereof, with calculations to convert from one to the other). In case of a window-based controller, FSE_R is a window, and all the text below should be considered to refer to window, not rates.

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 optionally (if the algorithm supports it) the desired rate. The desired rate is less than the calculated rate in case of application-limited flows; otherwise, it is the same as the calculated rate.

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 [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 [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. Application

This section specifies how the FSE can be applied to specific congestion control mechanisms and makes general recommendations that facilitate applying the FSE to future congestion controls.

6.1. NADA

Network-Assisted Dynamic Adapation (NADA) [nada] is a congestion control scheme for rtcweb. It calculates a reference rate R_n upon receiving an acknowledgment, and then, based on the reference rate, it calculates a video target rate R_v and a sending rate for the flows, R_s.

When applying the FSE to NADA, the UPDATE function call described in Section 5.3 gives the FSE NADA's reference rate R_n. The recommended algorithm for NADA is the Active FSE in Section 5.3.1. In step 3 (c), when the FSE_R(i) is "sent" to the flow i, this means updating R_v and R_s of flow i with the value of FSE_R(i).

NADA simulation results are available from http://heim.ifi.uio.no/safiquli/coupled-cc/. The next version of this document will refer to a technical report that will be made available at the same URL.

6.2. General recommendations

This section will provides general advice for applying the FSE to congestion control mechanisms. TEMPORARY NOTE: Future versions of this document will contain a longer list.

Receiver-side calculations:
: When receiver-side calculations make assumptions about the rate of the sender, the calculations need to be synchronized or the receiver needs to be updated accordingly. This applies to TFRC [RFC5348], for example, where simulations showed somewhat less favorable results when using the FSE without a receiver-side change [fse].

7. Acknowledgements

This document has benefitted from discussions with and feedback from David Hayes, Mirja Kuehlewind, Andreas Petlund, David Ros (who also gave the FSE its name), Zaheduzzaman Sarker and Varun Singh. The authors would like to thank Xiaoqing Zhu for helping with NADA.

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

8. IANA Considerations

This memo includes no request to IANA.

9. 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.

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.

[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.

10.2. Informative References

[fse] Islam, S., Welzl, M., Gjessing, S., and N. Khademi, "Coupled Congestion Control for RTP Media", ACM SIGCOMM Capacity Sharing Workshop (CSWS 2014); extended version available as a technical report from http://safiquli.at.ifi.uio.no/paper/fse-tech-report.pdf , 2014.

[nada] Zhu, X., Pan, R., Ramalho, M., Mena, S., Ganzhorn, C., Jones, P., and S. De Aronco, "NADA: A Unified Congestion Control Scheme for Real-Time Media", draft-ietf-rmcat-nada-00 (work in progress), April 2015.

[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-18.txt (work in progress), October 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.

[sbd] Hayes, D., Ferlin, S., and M. Welzl, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media", draft-ietf-rmcat-sbd-00.txt (work in progress), May 2015.

[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, when round-trip times of flows are unequal, shorter-RTT flows will update and react to the overall FSE state more often than longer-RTT flows, which can produce unwanted side effects. This problem is more significant when the congestion control convergence depends on the RTT. While the passive algorithm works better for congestion controls with RTT-independent convergence, it can still produce oscillations on short time scales. The algorithm described below is therefore considered as highly experimental.

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

- 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(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

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

B.2. Changes from -01 to -02

- Included an active version of the algorithm which is simpler.
- Replaced "greedy flow" with "bulk data transfer" and "non-greedy" with "application-limited".
- Updated new_CR to CC_R, and CR to FSE_R for better understanding.

B.3. Changes from -02 to -03

- 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.

- Moved the passive variant of the algorithm to appendix.

B.4. Changes from -03 to -04

- Extended SBD section.
- Added a note about window-based controllers.

B.5. Changes from -04 to -05

- Added a section about applying the FSE to specific congestion control algorithms, with a subsection specifying its use with NADA.

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

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2016

X. Zhu
S. Mena
Cisco Systems
Z. Sarker
Ericsson AB
July 3, 2015

Modeling Video Traffic Sources for RMCAT Evaluations
draft-zhu-rmcat-video-traffic-source-02

Abstract

This document describes two reference video traffic source models for evaluating RMCAT candidate algorithms. The first model statistically characterizes the behavior of a live video encoder in response to changing requests on target video rate. The second model is trace-driven, and emulates the encoder output by scaling the pre-encoded video frame sizes from a widely used video test sequence. Both models are designed to strike a balance between simplicity, repeatability, and authenticity in modeling the interactions between a video traffic source and the congestion control module.

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 4, 2016.

Table of Contents

1. Introduction

When evaluating candidate congestion control algorithms designed for real-time interactive media, it is important to account for the characteristics of traffic patterns generated from a live video encoder. Unlike synthetic traffic sources that can conform perfectly to the rate changing requests from the congestion control module, a live video encoder can be sluggish in reacting to such changes. Output rate of a live video encoder also typically deviates from the target rate due to uncertainties in the encoder rate control process. Consequently, end-to-end delay and loss performance of a real-time media flow can be further impacted by rate variations introduced by the live encoder.

On the other hand, evaluation results of a candidate RMCAT algorithm should mostly reflect performance of the congestion control module, and somewhat decouple from pecularities of any specific video codec. It is also desirable that evaluation tests are repeatable, and be easily duplicated across different candidate algorithms.

One way to strike a balance between the above considerations is to evaluate RMCAT algorithms using a synthetic video traffic source model that captures key characteristics of the behavior of a live video encoder. To this end, this draft presents two reference models. The first is based on statistical modelling; the second is trace-driven. The draft also discusses the pros and cons of each approach, as well as the possibility to combine both.

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 RFC2119 [RFC2119].

3. Desired Behavior of A Synthetic Video Traffic Model

A live video encoder employs encoder rate control to meet a target rate by varying its encoding parameters, such as quantization step size, frame rate, and picture resolution, based on its estimate of the video content (e.g., motion and scene complexity). In practice, however, several factors prevent the output video rate from perfectly conforming to the input target rate.

Due to uncertainties in the captured video scene, the output rate typically deviates from the specified target. In the presence of a significant change in target rate, it sometimes takes several frames before the encoder output rate converges to the new target. Finally, while most of the frames in a live session are encoded in predictive mode, the encoder can occasionally generate a large intra-coded frame (or a frame partially containing intra-coded blocks) in an attempt to recover from losses, to re-sync with the receiver, or during the transient period of responding to target rate or spatial resolution changes.

Hence, a synthetic video source should have the following capabilities:

- To change bitrate. This includes ability to change framerate and/or spatial resolution, or to skip frames when required.

- To fluctuate around the target bitrate specified by the congestion control module.

- To delay in convergence to the target bitrate.

- To generate intra-coded or repair frames on demand.

While there exists many different approaches in developing a synthetic video traffic model, it is desirable that the outcome follows a few common characteristics, as outlined below.

- Low computational complexity: The model should be computationally lightweight, otherwise it defeats the whole purpose of serving as a substitute for a live video encoder.

- Temporal pattern similarity: The individual traffic trace instances generated by the model should mimic the temporal pattern of those from a real video encoder.

- Statistical resemblance: The synthetic traffic should match the outcome of the real video encoder in terms of statistical characteristics, such as the mean, variance, peak, and autocorrelation coefficients of the bitrate. It is also important that the statistical resemblance should hold across different time scales, ranging from tens of milliseconds to sub-seconds.

- Wide range of coverage: The model should be easily configurable to cover a wide range of codec behaviors (e.g., with either fast or slow reaction time in live encoder rate control) and video content variations (e.g, ranging from high-motion to low-motion).

These distinct behavior features can be characterized via simple statistical models, or a trace-driven approach. We present an example of each in Section 5 and Section 6

4. Interactions Between Synthetic Video Traffic Source and Other Components at the Sender

Figure 1 depitcs the interactions of the synthetic video encoder with other components at the sender, such as the application, the congestion control module, the media packet transport module, etc. Both reference models, as described later in Section 5 and Section 6, follow the same set of interactions.

The synthetic video encoder takes in raw video frames captured by the camera and then dynamically generates a sequence of encoded video frames with varying size and interval. These encoded frames are processed by other modules in order to transmit the video stream over the network. During the lifetime of a video transmission session, the synthetic video encoder will typically be required to adapt its

encoding bitrate, and sometimes the spatial resolution and frame rate.

In our model, the synthetic video encoder module has group of incoming and outgoing interface calls that allow for interaction with other modules. The following are some of the possible incoming interface calls --- marked as (a) in Figure 1 --- that the synthetic video encoder may accept. The list is not exhaustive and can be complemented by other interface calls if deemed necessary.

- Target rate R_v(t): requested at time t, typically from the congestion control module. Depending on the congestion control algorithm in use, the update requests can either be periodic (e.g., once per 1 second), or on-demand (e.g., only when a drastic bandwidth change over the network is observed).

- Target frame rate FPS(t): the instantaneous frame rate measured in frames-per-second at time t. This depends on the native camera capture frame rate as well as the target/preferred frame rate configured by the application or user.

- Frame resolution XY(t): the 2-dimensional vector indicating the preferred frame resolution in pixels at time t. Several factors govern the resolution requested to the synthetic video encoder over time. Examples of such factors are the capturing resolution of the native camera; or the current target rate R_v(t), since very small resolutions do not make sense with very high bitrates, and vice-versa.

- Instant frame skipping: the request to skip the encoding of one or several captured video frames, for instance when a drastic decrease in available network bandwidth is detected.

- On-demand generation of intra (I) frame: the request to encode another I frame to avoid further error propagation at the receiver, if severe packet losses are observed. This request typically comes from the error control module.

An example of outgoing interface call --- marked as (b) in Figure 1 --- is the rate range, that is, the dynamic range of the video encoder's output rate for the current video contents: [R_min, R_max]. Here, R_min and R_max are meant to capture the dynamic rate range the encoder is capable of outputting. This typically depends on the

video content complexity and/or display type (e.g., higher R_max for video contents with higher motion complexity, or for displays of higher resolution). Therefore, these values will not change with R_v, but may change over time if the content is changing.

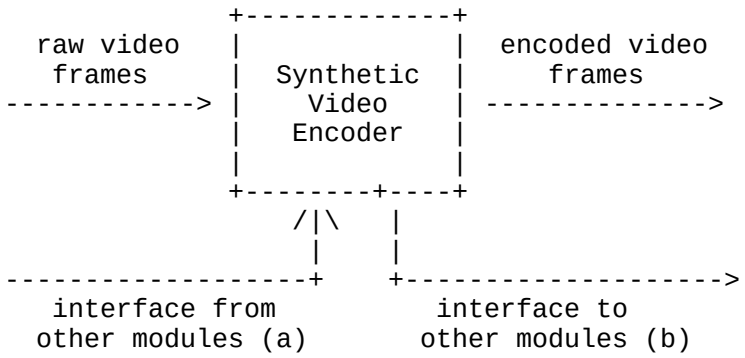


Figure 1: Interaction between synthetic video encoder, congestion control, and packet transport module.

5. A Statistical Reference Model

In this section, we describe one simple statistical model of the live video encoder traffic source. Figure 2 summarizes the list of tuable parameters in this statistical model. A more comprehensive survey of popular methods for modelling video traffic source behavior can be found in [Tanwir2013].

Notation	Parameter Name	Example Value
R_v(t)	Target rate request at time t	1 Mbps
R_o(t)	Output rate at time t	1.2 Mbps
tau_v	Encoder reaction latency	0.2 s
K_d	Burst duration during transient	5 frames
K_r	Burst size dur transient	5:1
R_e(t)	Error in output rate at time t	0.2 Mbps
SIGMA	standard deviation of normally distributed relative rate error	0.1
DELTA	upper and lower bound (+/-) of uniformally distributed relative rate error	0.1
R_min	minimum rate supported by video encoder or content activity	150 Kbps
R_max	maximum rate supported by video encoder or content activity	1.5Mbps

Figure 2: List of tunable parameters in a statistical video traffic source model.

5.1. Time-damped response to target rate update

While the congestion control module can update its target rate request R_v(t) at any time, our model dictates that the encoder will only react to such changes after tau_v seconds from a previous rate transition. In other words, when the encoder has reacted to a rate change request at time t, it will simply ignore all subsequent rate change requests until time t+tau_v.

5.2. Temporary burst/oscillation during transient

The output rate R_o during the period [t, t+tau_v] is considered to be in transient. Based on observations from video encoder output data, we model the transient behavior of an encoder upon reacting to a new target rate request in the form of largely varying output sizes. It is assumed that the overall average output rate R_o during this period matches the target rate R_v. Consequently, the occasional burst of large frames are followed by smaller-than average encoded frames.

This temporary burst is characterized by two parameters:

o burst duration K_d: number frames in the burst event; and

- burst size K_r: ratio of a burst frame and average frame size at steady state.

It can be noted that these burst parameters can also be used to mimic the insersion of a large on-demand I frame in the presence of severe packet losses. The values of K_d and K_r are fitted to reflect the typical ratio between I and P frames for a given video content.

5.3. Output rate fluctuation at steady state

We model output rate R_o as randomly fluctuating around the target rate R_v after convergence. There are two variants in modeling the random fluctuation R_e = R_o - R_v:

- As normal distribution: with a mean of zero and a standard deviation SIGMA specified in terms of percentage of the target rate. A typical value of SIGMA is 10 percent of target rate.

- As uniform distribution bounded between -DELTA and DELTA. A typical value of DELTA is 10 percent of target rate.

The distribution type (normal or uniform) and model parameters (SIGMA or DELTA) can be learned from data samples gathered from a live encoder output.

5.4. Rate range limit imposed by video content

The output rate R_o is further clipped within the dynamic range [R_min, R_max], which in reality are dictated by scene and motion complexity of the captured video content. In our model, these parameters are specified by the application.

6. A Trace-Driven Model

We now present the second approach to model a video traffic source. This approach is based on running an actual live video encoder offline on a set of chosen raw video sequences and using the encoder’s output traces for constructing a synthetic live encoder. With this approach, the recorded video traces naturally exhibit temporal fluctuations around a given target rate request R_v(t) from the congestion control module.

The following list summarizes this approach’s main steps:

1) Choose one or more representative raw video sequences.

2) Using an actual live video encoder, encode the sequences at various bitrates. Keep just the sequences of frame sizes for each bitrate.

3) Construct a data structure that contains the output of the previous step. The data structure should allow for easy bitrate lookup.

4) Upon a target bitrate request R_v(t) from the controller, look up the closest bitrates among those previously stored. Use the frame size sequences stored for those bitrates to approximate the frame sizes to output.

5) The output of the synthetic encoder contains "encoded" frames with random contents but with realistic sizes.

Section 6.1 explains steps 1), 2), and 3), Section 6.2 elaborates on steps 4) and 5). Finally, Section 6.3 briefly discusses the possibility to extend the model for supporting variable frame rate and/or variable frame resolution.

6.1. Choosing the video sequence and generating the traces

The first step we need to perform is a careful choice of a set of video sequences that are representative of the use cases we want to model. Our use case here is video conferencing, so we must choose a low-motion sequence that resembles a "talking head", for instance a news broadcast or a video capture of an actual conference call.

The length of the chosen video sequence is a tradeoff. If it is too long, it will be difficult to manage the data structures containing the traces we will produce in the next steps. If it is too short, there will be an obvious periodic pattern in the output frame sizes, leading to biased results when evaluating congestion controller performance. In our experience, a one-minute-long sequence is a fair tradeoff.

Once we have chosen the raw video sequence, denoted S, we use a live encoder, e.g. [H264] or [HEVC] to produce a set of encoded sequences. As discussed in Section 3, a live encoder’s output bitrate can be tuned by varying three input parameters, namely, quantization step size, frame rate, and picture resolution. In order to simplify the choice of these parameters for a given target rate, we assume a fixed frame rate (e.g. 25 fps) and a fixed resolution (e.g., 480p). See section 6.3 for a discussion on how to relax these assumptions.

Following these simplifications, we run the chosen encoder by setting a constant target bitrate at the beginning, then letting the encoder vary the quantization step size internally while encoding the input video sequence. Besides, we assume that the first frame is encoded as an I-frame and the rest are P-frames. We further assume that the encoder algorithm does not use knowledge of frames in the future so as to encode a given frame.

We define R_min and R_max as the minimum and maximum bitrate at which the synthetic codec is to operate. We divide the bitrate range between R_min and R_max in n_s + 1 bitrate steps of length l = (R_max - R_min) / n_s. We then use the following simple algorithm to encode the raw video sequence.

```
r = R_min
while r <= R_max do
    Traces[r] = encode_sequence(S, r, e)
    r = r + l
```

where function encode_sequence takes as parameters, respectively, a raw video sequence, a constant target rate, and an encoder algorithm; it returns a vector with the sizes of frames in the order they were encoded. The output vector is stored in a map structure called Traces, whose keys are bitrates and values are frame size vectors.

The choice of a value for n_s is important, as it determines the number of frame size vectors stored in map Traces. The minimum value one can choose for n_s is 1, and its maximum value depends on the amount of memory available for holding map Traces. A reasonable value for n_s is one that makes the steps' length l = 200 kbps. We will further discuss step length l in the next section.

6.2. Using the traces in the syntethic codec

The main idea behind the trace-based synthetic codec is that it mimics a real live codec's rate adaptation when the congestion controller updates the target rate R_v(t). It does so by switching to a different frame size vector stored in the map Traces when needed.

6.2.1. Main algorithm

We maintain two variables r_current and t_current:

* r_current points to one of the keys of the map Traces. Upon a change in the value of R_v(t), typically because the congestion controller detects that the network conditions have changed, r_current is updated to the greatest key in Traces that is less than

or equal to the new value of R_v(t). For the moment, we assume the value of R_v(t) to be clipped in the range [R_min, R_max].

```
r_current = r
such that
   ( r in keys(Traces)  and
   r <= R_v(t)  and
(not(exists) r' in keys(Traces) such that r < r' <= R_v(t)) )
```

* t_current is an index to the frame size vector stored in Traces[r_current]. It is updated every time a new frame is due. We assume all vectors stored in Traces to have the same size, denoted size_traces. The following equation governs the update of t_current:

```
if t_current < SkipFrames then
    t_current = t_current + 1
else
    t_current = ((t_current+1-SkipFrames) % (size_traces- SkipFrames))
                + SkipFrames
```

where operator % denotes modulo, and SkipFrames is a predefined constant that denotes the number of frames to be skipped at the beginning of frame size vectors after t_current has wrapped around. The point of constant SkipFrames is avoiding the effect of periodically sending a (big) I-frame followed by several smaller-than-normal P-frames. We typically set SkipFrames to 20, although it could be set to 0 if we are interested in studying the effect of sending I-frames periodically.

We initialize r_current to R_min, and t_current to 0.

When a new frame is due, we need to calculate its size. There are three cases:

a) R_min <= R_v(t) < Rmax: In this case we use linear interpolation of the frame sizes appearing in Traces[r_current] and Traces[r_current + l]. The interpolation is done as follows:

```
size_lo = Traces[r_current][t_current]
    size_hi = Traces[r_current + l][t_current]
    distance_lo = ( R_v(t) - r_current ) / l
    framesize = size_hi * distance_lo + size_lo * (1 - distance_lo)
```

b) R_v(t) < R_min: In this case, we scale the trace sequence with the lowest bitrate, in the following way:

```
factor = R_v(t) / R_min
    framesize = max(1, factor * Traces[R_min][t_current])
```

c) R_v(t) >= R_max: We also use scaling for this case. We use the trace sequence with the greatest bitrate:

```
factor = R_v(t) / R_max
    framesize = factor * Traces[R_max][t_current]
```

In case b), we set the minimum to 1 byte, since the value of factor can be arbitrarily close to 0.

6.2.2. Notes to the main algorithm

* Reacting to changes in target bitrate. Similarly to the statistical model presented in Section 5, the trace-based synthetic codec has a time bound, tau_v, to reacting to target bitrate changes. If the codec has reacted to an update in R_v(t) at time t, it will delay any further update to R_v(t) to time t + tau_v. Note that, in any case, the value of tau_v cannot be chosen shorter than the time between frames, i.e. the inverse of the frame rate.

* I-frames on demand. The synthetic codec could be extended to simulate the sending of I-frames on demand, e.g., as a reaction to losses. To implement this extension, the codec's API is augmented with a new function to request a new I-frame. Upon calling such function, t_current is reset to 0.

* Variable length l of steps defined between R_min and R_max. In the main algorithm's description, the step length l is fixed. However, if the range [R_min, R_max] is very wide, it is also possible to define a set of steps with a non-constant length. The idea behind this modification is that the difference between 400 kbps and 600 kbps as bitrate is much more important than the difference between 4400 kbps and 4600 kbps. For example, one could define steps of length 200 Kbps under 1 Mbps, then length 300 kbps between 1 Mbps and 2 Mbps, 400 kbps between 2 Mbps and 3 Mbps, and so on.

6.3. Varying frame rate and resolution

The trace-based synthetic codec model explained in this section is relatively simple because we have fixed the frame rate and the frame resolution. The model could be extended to have variable frame rate, variable spatial resolution, or both.

When the encoded picture quality at a given bitrate is low, one can potentially decrease the frame rate (if the video sequence is currently in low motion) or the spatial resolution in order to

improve quality-of-experince (QoE) in the overall encoded video. On the other hand, if target bitrate increases to a point where there is no longer a perceptible improvement in the picture quality of individual frames, then one might afford to increase the spatial resolution or the frame rate (useful if the video is currently in high motion).

Many techniques have been proposed to choose over time the best combination of encoder quatization step size, frame rate, and spatial resolution in order to maximize the quality of live video codecs [Ozer2011][Hu2010]. Future work may consider extending the trace-based codec to accommodate variable frame rate and/or resolution.

From the perspective of congestion control, varying the spatial resolution typically requires a new intra-coded frame to be generated, thereby incurring a temporary burst in the output traffic pattern. The impact of frame rate change tends to be more subtle: reducing frame rate from high to low leads to sparsely spaced larger encoded packets instead of many densely spaced smaller packets. Such difference in traffic profiles may still affect the performance of congestion control, especially when outgoing packets are not paced at the transport module. We leave the investigation of varying frame rate to future work.

7. Comparing and Combining The Two Models

It is worthwhile noting that the statistical and trace-based models each has its own advantages and drawbacks. Both models are fairly simple to implement. However, it takes significantly more effort to fit the parameters of a statistical model to actual encoder output data whereas trace-based models does not require such fitting. On the other hand, once validated, the statistical model is more flexible in mimicking a wide range of encoder/content behavior by simply varying the correponding parameters in the model. In contrast, a trace-driven model relies, by definition, on additional data collection efforts for accommodating new codecs or video contents.

In general, trace-based model is more realistic for mimicking ongoing, steady-state behavior of a video traffic source whereas statistical model is more versatile for simulating transient events (e.g., when target rate changes from A to B with temporary bursts during the transition). Therefore, it may be desirable to combine both approaches into a hybrid model, using traces for steady-state and statistical model for transients.

8. Implementation Status

The statistical model has been implemented as a traffic generator module within the [ns-2] network simulation platform. The trace-driven model has been implemented as a stand-alone traffic source module which can be easily integrated into the [ns-3] network simulation platform.

Authors of this draft are currently in the process of providing open source access to both implementations.

9. IANA Considerations

There are no IANA impacts in this memo.

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.

[H264] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services", <http://www.itu.int/rec/T-REC-H.264-201304-I>.

[HEVC] ITU-T Recommendation H.265, "High efficiency video coding", .

10.2. Informative References

[Hu2010] Hu, H., Ma, Z., and Y. Wang, "Optimization of Spatial, Temporal and Amplitude Resolution for Rate-Constrained Video Coding and Scalable Video Adaptation", inproceedings in Proc. 19th IEEE International Conference on Image Processing (ICIP’12), September 2012.

[Ozer2011] Ozer, J., "Video Compression for Flash, Apple Devices and HTML5", ISBN ISBN-13:978-0976259503, 2011.

[Tanwir2013] Tanwir, S. and H. Perros, "A Survey of VBR Video Traffic Models", journal IEEE Communications Surveys and Tutorials, vol. 15, no. 5, pp. 1778-1802., October 2013.

[ns-2] "The Network Simulator - ns-2", <http://www.isi.edu/nsnam/ns/>.

[ns-3] "The Network Simulator - ns-3", <https://www.nsnam.org/>.

Authors' Addresses

Xiaoqing Zhu
Cisco Systems
12515 Research Blvd., Building 4
Austin, TX 78759
USA

Email: xiaoqzhu@cisco.com

Sergio Mena de la Cruz
Cisco Systems
EPFL, Quartier de l'Innovation, Batiment E
Ecublens, Vaud 1015
Switzerland

Email: semena@cisco.com

Zaheduzzaman Sarker
Ericsson AB
Luleae, SE 977 53
Sweden

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