

RMCAWG
Internet-Draft
Intended status: Informational
Expires: April 23, 2014

V. Singh
J. Ott
Aalto University
October 20, 2013

Evaluating Congestion Control for Interactive Real-time Media
draft-singh-rmcat-cc-eval-04

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Metrics	3
3.1. RTP Log Format	5
4. Guidelines	5
4.1. Avoiding Congestion Collapse	5
4.2. Stability	5
4.3. Media Traffic	6
4.4. Start-up Behaviour	6
4.5. Diverse Environments	6
4.6. Varying Path Characteristics	7
4.7. Reacting to Transient Events or Interruptions	7
4.8. Fairness With Similar Cross-Traffic	7
4.9. Impact on Cross-Traffic	7
4.10. Extensions to RTP/RTCP	8
5. Minimum Requirements for Evaluation	8
6. Evaluation Parameters	8
6.1. Bottleneck Traffic Flows	8
6.2. Access Links	9
6.3. Example Bottleneck Link Parameters	9
6.4. DropTail Router Queue Parameters	10
6.5. Media Flow Parameters	11
6.6. Cross-traffic Parameters	11
7. Status of Proposals	11
8. Security Considerations	12
9. IANA Considerations	12
10. Contributors	12
11. Acknowledgements	12
12. References	12
12.1. Normative References	12
12.2. Informative References	13
Appendix A. Application Trade-off	14
A.1. Measuring Quality	14
Appendix B. Proposal to evaluate Self-fairness of RMCAT congestion control algorithm	14
B.1. Evaluation Parameters	15
B.1.1. Media Traffic Generator	15
B.1.2. Bottleneck Link Bandwidth	16
B.1.3. Bottleneck Link Queue Type and Length	16
B.1.4. RMCAT flows and delay legs	16
B.1.5. Impairment Generator	17
B.2. Proposed Passing Criteria	17
B.3. Extensibility of the Experiment	17
Appendix C. Change Log	18
C.1. Changes in draft-singh-rmcat-cc-eval-04	18
C.2. Changes in draft-singh-rmcat-cc-eval-03	18

C.3. Changes in draft-singh-rmcat-cc-eval-02	18
C.4. Changes in draft-singh-rmcat-cc-eval-01	18
Authors' Addresses	19

1. Introduction

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

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

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

2. Terminology

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

3. Metrics

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

- o Throughput.
- o Minimizing oscillations in the transmission rate (stability) when the end-to-end capacity varies slowly.
- o Delay.
- o Reactivity to transient events.

- o Packet losses and discards.
- o Section 2.1 of [RFC5166] discusses the tradeoff between throughput, delay and loss.

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

1. Sending rate, Receiver rate, Goodput
2. Packet delay
3. Packet loss
4. If using, retransmission or FEC: residual loss
5. Packets discarded from the playout or de-jitter buffer

[Open issue (1): The "unfairness" test is (measured at 1s intervals):

1. Do not trigger the circuit breaker.
2. Over 3 times or less than 1/3 times the throughput for an RMCAT media stream compared to identical RMCAT streams competing on a bottleneck, for a case when the competing streams have similar RTTs.
3. Over 3 times delay compared to RTT measurements performed before starting the RMCAT flow or for the case when competing with identical RMCAT streams having similar RTTs.

]

[Open issue (2): Possibly using Jain-fairness index.]

Convergence time: the time taken to reach a stable rate at startup, after the available link capacity changes, or when new flows get added to the bottleneck link.

Bandwidth Utilization, defined as ratio of the instantaneous sending rate to the instantaneous bottleneck capacity. This metric is useful when an RMCAT flow is by itself or competing with similar cross-traffic.

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

3.1. RTP Log Format

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

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

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

4. Guidelines

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

4.1. Avoiding Congestion Collapse

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

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

4.2. Stability

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

4.3. Media Traffic

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

The evaluation may be done in two stages. In the first stage, the endpoint generates traffic at the rate calculated by the congestion controller. In the second stage, real codecs or models of video codecs are used to mimic application-limited data periods and varying video frame sizes.

4.4. Start-up Behaviour

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

[Editor's note: requires a robust definition for unfriendliness and convergence time.]

4.5. Diverse Environments

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

4.6. Varying Path Characteristics

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

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

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

4.7. Reacting to Transient Events or Interruptions

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

4.8. Fairness With Similar Cross-Traffic

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

[Editor's note: If we define Unfriendliness then that criteria should be applied here.]

4.9. Impact on Cross-Traffic

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

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

4.10. Extensions to RTP/RTCP

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

5. Minimum Requirements for Evaluation

[Editor's Note: If needed, a minimum evaluation criteria can be based on the above guidelines or defined tests/scenarios.]

6. Evaluation Parameters

An evaluation scenario is created from a list of network, link and flow characteristics. The example parameters discussed in the following subsections are meant to aid in creating evaluation scenarios and do not describe an evaluation scenario. The scenario discussed in Appendix B takes into account all these parameters.

6.1. Bottleneck Traffic Flows

The network scenario describes the types of flows sharing the common bottleneck with a single RMCAT flow, they are:

1. A single RMCAT flow by itself.
2. Competing with similar RMCAT flows. These competing flows may use the same algorithm or another candidate RMCAT algorithm.
3. Compete with long-lived TCP.
4. Compete with bursty TCP.
5. Compete with LEDBAT flows.
6. Compete with unresponsive interactive media flows (i.e., not only CBR).

Figure 1 shows an example evaluation topology, where S1..Sn are traffic sources, these sources are either RMCAT or a mixture of traffic flows listed above. R1..Rn are the corresponding receivers. A and B are routers that can be configured to introduce impairments. Access links are in between the sender/receiver and the router, while the bottleneck link is between the Routers A and B.

```

+---+ Access                               Access +---+
|S1 |===== \                             / =====|R1 |
+---+ link  \\\                           // link  +---+

```

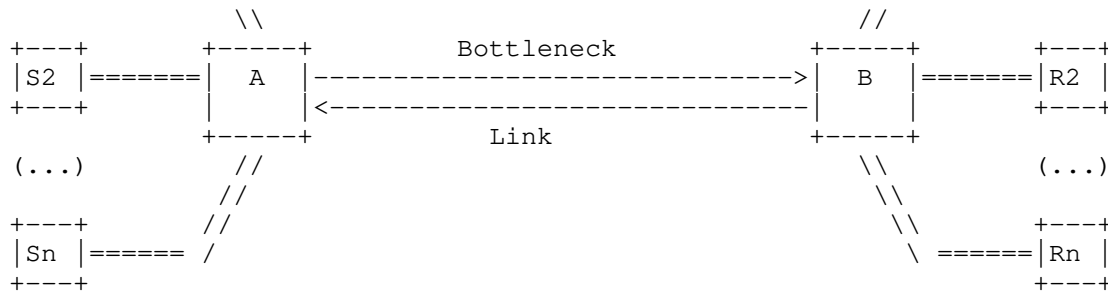



Figure 1: Simple Topology

[Open Issue: Discuss more complex topologies]

6.2. Access Links

The media senders and receivers are typically connected to the bottleneck link, common access links are:

1. Ethernet (LAN)
2. Wireless LAN (WLAN)
3. 3G/LTE

[Open issue: point to a reference containing parameters or traces to model WLAN and 3G/LTE.]

A real-world network typically consists of a mixture of links, the most important aspect is to identify the location of the bottleneck link. The bottleneck link can move from one node to another depending on the amount of cross-traffic or due to the varying link capacity. The design of the experiments should take this into account. In the simplest case the access link may not be the bottleneck link but an intermediate node.

6.3. Example Bottleneck Link Parameters

The bottleneck link carries multiple flows, these flows may be other RMCAT flows or other types of cross-traffic. The experiments should dimension the bottleneck link based on the number of flows and the expected behavior. For example, if 5 media flows are expected to share the bottleneck link equally, the bottleneck link is set to 5 times the desired transmission rate.

If the experiment carries only media in one direction, then the upstream (sender to receiver) bottleneck link carries media packets

while the downstream (receiver to sender) bottleneck carries the feedback packets. The bottleneck link parameters discussed in this section apply only to a single direction, hence the bottleneck link in the reverse direction can choose the same or have different parameters.

The link latency corresponds to the propagation delay of the link, i.e., the time it takes for a packet to traverse the bottleneck link, it does not include queuing delay. In an experiment with several links the experiment should describe if the links add latency or not. It is possible for experiments to have multiple hops with different link latencies. Experiments are expected to verify that the congestion control is able to work in challenging situations, for example over trans-continental and/or satellite links. The experiment should pick link latency values from the following:

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

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

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

These fractional losses can be generated using traces, Gilbert-Elliot model, randomly (uncorrelated) loss.

6.4. DropTail Router Queue Parameters

The router queue length is measured as the time taken to drain the FIFO queue, they are:

1. QoS-aware (or short): 70ms

2. Nominal: 500ms

3. Buffer-bloated: 2000ms

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

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

6.5. Media Flow Parameters

The media sources can be modeled in two ways. In the first, the sources always have data to send, i.e., have no data limited intervals and are able to generate the media rate requested by the RMCAT congestion control algorithm. In the second, the traffic generator models the behavior of a media codec, mainly the burstiness (time-varying data produced by a video GOP).

At the beginning of the session, the media sources are configured to start at a given start rate, they are:

1. 200 kbps
2. 800 kbps
3. 1300 kbps
4. 4000 kbps

6.6. Cross-traffic Parameters

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

[Open issue: short-lived/bursty TCP cross-traffic parameters are still TBD.

7. Status of Proposals

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

[Editor's Note: Should there be a distinction between "Informational" and "Experimental" drafts for congestion control algorithms in RMCAT. [RFC5033] describes Informational proposals as algorithms that are not safe for deployment but are proposals to experiment with in simulation/testbeds. While Experimental algorithms are ones that are deemed safe in some environments but require a more thorough evaluation (from the community).]

8. Security Considerations

Security issues have not been discussed in this memo.

9. IANA Considerations

There are no IANA impacts in this memo.

10. Contributors

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

Michael Ramalho provided the text for the scenario discussed in Appendix B.

11. Acknowledgements

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

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

12. References

12.1. Normative References

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

- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [I-D.jesup-rmcat-reqs]
Jesup, R., "Congestion Control Requirements For RMCAT", draft-jesup-rmcat-reqs-01 (work in progress), February 2013.
- [I-D.ietf-avtcore-rtp-circuit-breakers]
Perkins, C. and V. Singh, "RTP Congestion Control: Circuit Breakers for Unicast Sessions", draft-ietf-avtcore-rtp-circuit-breakers-01 (work in progress), October 2012.

12.2. Informative References

- [RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, August 2007.
- [RFC5166] Floyd, S., "Metrics for the Evaluation of Congestion Control Mechanisms", RFC 5166, March 2008.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [SA4-EVAL]
R1-081955, 3GPP., "LTE Link Level Throughput Data for SA4 Evaluation Framework", 3GPP R1-081955, 5 2008.
- [SA4-LR]
S4-050560, 3GPP., "Error Patterns for MBMS Streaming over UTRAN and GERAN", 3GPP S4-050560, 5 2008.
- [TCP-eval-suite]
Lachlan, A., Marcondes, C., Floyd, S., Dunn, L., Guillier, R., Gang, W., Eggert, L., Ha, S., and I. Rhee, "Towards a Common TCP Evaluation Suite", Proc. PFLDnet. 2008, August 2008.

Appendix A. Application Trade-off

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

A.1. Measuring Quality

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

Appendix B. Proposal to evaluate Self-fairness of RMCAT congestion control algorithm

The goal of the experiment discussed in this section is to initially take out as many unknowns from the scenario. Later experiments can define more complex environments, topologies and media behavior. This experiment evaluates the performance of the RMCAT sender competing with other similar RMCAT flows (running the same algorithm or other RMCAT proposals) on the bottleneck link. There are up to 20 RMCAT flows competing for capacity, but the media only flows in one direction, from senders (S1..S20) to receivers (R1..R20) and the feedback packets flow in the reverse direction.

Figure 2 shows the experiment setup and it has subtle differences compared to the simple topology in Figure 1. Groups of 10 receivers are connected to the bottleneck link through two different routers (Router C and D). The rationale for adding these additional routers is to create two delay legs, i.e., two groups of endpoints with different network latencies and measure the performance of the RMCAT congestion control algorithm. If fewer than 10 sources are initialized, all traffic flows experience the same delay because they share the same delay leg.

Router A has a single forward direction bottleneck link (i.e., the bottleneck capacity and delay constraints applies only to the media packets going from the sender to the receiver, the feedback packets are unaffected). Hence, the Round-Trip Time (RTT) is primarily composed of the bottleneck queue delay and any forward path (propagation) latency. The main reason for not applying any constraints on the return path is to provide the best-case performance scenario for the congestion control algorithm. In later experiments, it is possible to add similar capacity and delay constraints on the return path.

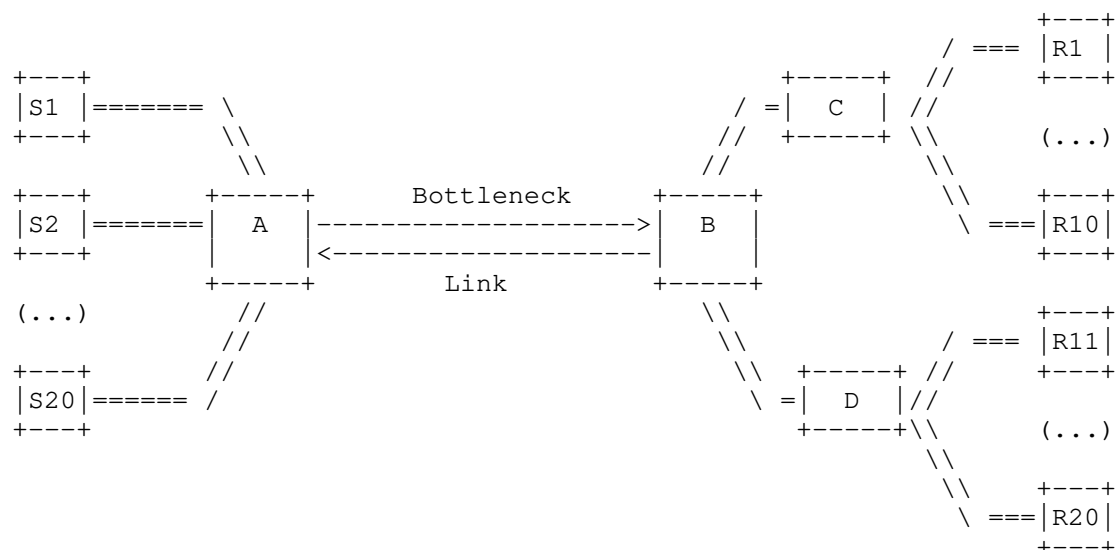


Figure 2: Self-fairness Evaluation Setup

Loss impairments are applied at Router C and Router D, but only to the feedback flows. If the losses are set to 0%, it represents a case where the return path is over-provisioned for all traffic. In later experiments the loss impairments can be added to the media path as well.

The media sources are configured to send infinite amount of data, i.e., the sources always have data to send and have no data limited intervals. Additionally, the media sources are always successful in generating the media rate requested by the RMCAT congestion control algorithm. In this experiment, we avoid the potentially complicated scenario of using media traffic generators that try to model the behavior of media codecs (mainly the burstiness).

B.1. Evaluation Parameters

B.1.1. Media Traffic Generator

The media source always generates at the rate requested by the congestion control and has infinite data to send. Furthermore, the media packet generator is subject to the following constraints:

1. It MUST emit a packet at least once per 100 ms time interval.
2. For low media rate source: when generating data at a rate less than a maximum length MTU every 100 ms would allow (e.g., 120

kbps = 1500 bytes/packet * 10 packets/sec * 8 bits/byte), the RMCAT source must modulate the packet size (RTP payload size) of RTP packets that are sent every 100 ms to attain the desired rate.

3. For high media rate sources: when generating data at a rate greater than a maximum length MTU every 100 ms would allow, the source must do so by sending (approximately) maximum MTU sized packets and adjusting the inter-departure interval to be approximately equal. The intent of this is to ensure the data is sent relatively smoothly independent of the bit rate, subject to the first constraint.

B.1.2. Bottleneck Link Bandwidth

The bottleneck link capacity is dimensioned such that each RMCAT flow in an ideal situation with perfectly equal capacity sharing for all the flows on the bottleneck obtains the following throughputs: 200 kbps, 800 kbps, 1.3 Mbps and 4 Mbps.

For example, experiments with five RMCAT flows with an 800 kbps/flow target rate should set the bottleneck link capacity to 4 Mbps.

B.1.3. Bottleneck Link Queue Type and Length

The bottleneck link queue (Router A) is a simple FIFO queue having a buffer length corresponding to 70 ms, 500 ms or 2000 ms (defined in Section 6.4) of delay at the bottleneck link rate (i.e., actual buffer lengths in bytes are dependent on bottleneck link bandwidth).

B.1.4. RMCAT flows and delay legs

Experiments run with 1, 3, 5, 10 and 20 RMCAT sources, they are outlined as follows:

1. Experiments with 1, 3, and 5 RMCAT flows, all RMCAT flows commence simultaneously. A single delay leg is used and the link latency is set to one of the following : 0 ms, 50 ms and 150 ms.
2. For 10 and 20 source experiments where all RMCAT flows begin simultaneously the sources are split evenly into two different bulk delay legs. One leg is set to 0 ms bulk delay leg and the other is set to 150 ms.
3. For 10 and 20 source experiments where the first set will use 0 ms of bulk delay and the second set will use 150 ms bulk delay.
 1. Random starts within interval [0 ms, 500 ms].

2. One "early-coming" flow (i.e., the 1st flow starting and achieving steady-state before the next N-1 simultaneously begin).
3. One "late-coming" flow (i.e., the Nth flow starting after steady-state has occurred for the existing N-1 flows).

These cases assess if there are any early or late-comer advantages or disadvantages for a particular algorithm and to see if any unfairness is reproducible or unpredictable.

[Open issue (A.1): which group does the early and late flow belong to?]

[Open issue (A.2): Start rate for the media flows]

B.1.5. Impairment Generator

Packet loss is created in the reverse path (affects only feedback packets). Cases of 0%, 1%, 5% and 10% are studied for the 1, 3, and 5 RMCAT flow experiments, losses are not applied to flows with 10 or 20 RMCAT flows.

B.2. Proposed Passing Criteria

[Editor's note: there has been little or no discussion on the below criteria, however, they are listed here for the sake of completeness.]

No unfairness is observed, i.e., at steady state each flow attains a throughput between $[B/(3*N), (3*B)/N]$, where B is the link bandwidth and N is the number of flows.

No flow experiences packet loss when queue length is set to 500 ms or greater.

All individual sources must be in their steady state within twenty LRTTs (where LRTT is defined as the RTT associated with the flow with the Largest RTT in the experiment).]

B.3. Extensibility of the Experiment

The above scenario describes only RMCAT sources competing for capacity on the bottleneck link, however, future experiments can use different types of cross-traffic (as described in Section 6.1).

Currently, the forward path (carrying media packets) is characterized to add delay and a fixed bottleneck link capacity, in the future packet losses and capacity changes can be applied to mimic a wireless

link layer (for e.g., WiFi, 3G, LTE). Additionally, only losses are applied to the reverse path (carrying feedback packets), later experiments can apply the same forward path (carrying media packets) impairments to the reverse path.

Appendix C. Change Log

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

C.1. Changes in draft-singh-rmcat-cc-eval-04

- o Incorporate feedback from IETF 87, Berlin.
- o Clarified metrics: convergence time, bandwidth utilization.
- o Changed fairness criteria to fairness test.
- o Added measuring pre- and post-repair loss.
- o Added open issue of measuring video quality to appendix.
- o clarified use of DropTail and AQM.
- o Updated text in "Minimum Requirements for Evaluation"

C.2. Changes in draft-singh-rmcat-cc-eval-03

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

C.3. Changes in draft-singh-rmcat-cc-eval-02

- o Added scenario descriptions.

C.4. Changes in draft-singh-rmcat-cc-eval-01

- o Removed QoE metrics.
- o Changed stability to steady-state.
- o Added measuring impact against few and many flows.

- o Added guideline for idle and data-limited periods.
- o Added reference to TCP evaluation suite in example evaluation scenarios.

Authors' Addresses

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

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

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

Email: jo@comnet.tkk.fi

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

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Definitions	3
3. Limitations	4
4. Architectural overview	5
5. Roles	6
5.1. SBD	6
5.2. FSE	6
5.3. Flows	7
5.3.1. Example algorithm 1 - Active FSE	7
5.3.2. Example algorithm 2 - Conservative Active FSE	8
6. Application	10
6.1. NADA	10
6.2. General recommendations	10
7. Acknowledgements	11
8. IANA Considerations	11
9. Security Considerations	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Appendix A. Example algorithm - Passive FSE	12
A.1. Example operation (passive)	15
Appendix B. Change log	19
B.1. Changes from -00 to -01	19
B.2. Changes from -01 to -02	19
B.3. Changes from -02 to -03	19
B.4. Changes from -03 to -04	20
B.5. Changes from -04 to -05	20
Authors' Addresses	20

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).

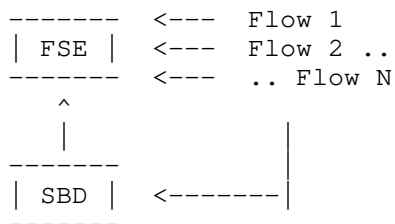


Figure 1: Coupled congestion control architecture

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:

- o a unique flow number to identify the flow
- o the FGI of the FG that it belongs to (based on the definitions in this document, a flow has only one bottleneck, and can therefore be in only one FG)
- o a priority P, which here is assumed to be represented as a floating point number in the range from 0.1 (unimportant) to 1 (very important). A negative value is used to indicate that a flow has terminated
- o The rate used by the flow 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 Adaption (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/safiquili/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://safiquili.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:

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

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

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


```

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

```

- (b) It updates S_CR, FSE_R(f) and DR(f).

```

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

```

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

```

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

```

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

```

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

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

```

- (e) It updates DR(f) and FSE_R(f) with Rate.

```

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

```

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

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

A.1. Example operation (passive)

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

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

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

S_CR = 1, TLO = 0

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

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

S_CR = 10, TLO = 0

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

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

S_CR = 11, TLO = 0

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

CC_R = 8. new_DR = infinity.

3 a) new_S_CR = 11; DELTA = 8 - 10 = -2.

3 b) FSE_R(f) = 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 = $\min(\text{infinity}, 0.5/1.5 * 10 + 0) = 3.33$.

3 e) DR(f) = FSE_R(f) = 3.33.

The resulting FSE looks as follows:

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

S_CR = 10, TLO = 0

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

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

The resulting FSE looks as follows:

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

S_CR = 11, TLO = 5.33

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

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

The resulting FSE looks as follows:

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

S_CR = 12, TLO = 0

Now, the total rate of the two flows is 2 + 9.33 = 11.33 Mbit/s. Finally, flow #1 terminates. It sets P to -1 and DR to 0. Let us

assume that it terminated late enough for flow #2 to still experience the network in a congested state, i.e. flow #2 decreases its rate in the next iteration.

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

The resulting FSE looks as follows:

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

S_CR = 9.33, TLO = 0

Appendix B. Change log

B.1. Changes from -00 to -01

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

B.2. Changes from -01 to -02

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

B.3. Changes from -02 to -03

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

- o Moved the passive variant of the algorithm to appendix.

B.4. Changes from -03 to -04

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

B.5. Changes from -04 to -05

- o 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: September 27, 2015

X. Zhu, R. Pan
M. A. Ramalho, S. Mena
C. Ganzhorn, P. E. Jones
Cisco Systems
S. De Aronco
Ecole Polytechnique Federale de Lausanne
March 26, 2015

NADA: A Unified Congestion Control Scheme for Real-Time Media
draft-zhu-rmcat-nada-06

Abstract

Network-Assisted Dynamic Adaptation (NADA) is a novel congestion control scheme for interactive real-time media applications, such as video conferencing. In NADA, the sender regulates its sending rate based on either implicit or explicit congestion signaling in a consistent manner. As one example of explicit signaling, NADA can benefit from explicit congestion notification (ECN) markings from network nodes. It also maintains consistent sender behavior in the absence of explicit signaling by reacting to queuing delay and packet loss.

This document describes the overall system architecture for NADA, as well as recommended behavior at the sender and the receiver.

Status of this Memo

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

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

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

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

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

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. System Model	3
4. NADA Receiver Behavior	4
4.1 Estimation of one-way delay and queuing delay	4
4.2 Estimation of packet loss/marketing ratio	5
4.3 Non-linear warping of delay	6
4.4 Aggregating congestion signals	7
4.5 Estimating receiving rate	7
4.6 Sending periodic feedback	7
4.7 Discussions on delay metrics	8
5. NADA Sender Behavior	9
5.1 Reference rate calculation	10
5.1.1 Accelerated ramp up	10
5.1.2. Gradual rate update	11
5.2 Video encoder rate control	12
5.3 Rate shaping buffer	12
5.4 Adjusting video target rate and sending rate	12
6. Incremental Deployment	13
7. Implementation Status	13
8. IANA Considerations	14
9. References	14
9.1 Normative References	14
9.2 Informative References	14
Appendix A. Network Node Operations	15
A.1 Default behavior of drop tail	16
A.2 ECN marking	16
A.3 PCN marking	16
Authors' Addresses	17

1. Introduction

Interactive real-time media applications introduce a unique set of challenges for congestion control. Unlike TCP, the mechanism used for real-time media needs to adapt quickly to instantaneous bandwidth changes, accommodate fluctuations in the output of video encoder rate control, and cause low queuing delay over the network. An ideal scheme should also make effective use of all types of congestion signals, including packet loss, queuing delay, and explicit congestion notification (ECN) [RFC3168] markings.

Based on the above considerations, this document describes a scheme called network-assisted dynamic adaptation (NADA). The NADA design benefits from explicit congestion control signals (e.g., ECN markings) from the network, yet also operates when only implicit congestion indicators (delay and/or loss) are available. In addition, it supports weighted bandwidth sharing among competing video flows.

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

2. Terminology

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

3. System Model

The overall system consists of the following elements:

- * Source media stream, in the form of consecutive raw video frames and audio samples;
- * Media encoder with rate control capabilities. It takes the source media stream and encodes it to an RTP stream at a target bit rate R_v . Note that the actual output rate from the encoder R_o may fluctuate around the target R_v . Also, the encoder can only change its rate at rather coarse time intervals, e.g., once every 0.5 seconds.
- * RTP sender, responsible for calculating the target bit rate R_n based on network congestion indicators (delay, loss, or ECN marking reports from the receiver), for updating the video encoder with a new target rate R_v , and for regulating the

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

* RTP receiver, responsible for measuring and estimating end-to-end delay based on sender RTP timestamp. In the presence of packet loss and ECN markings, it keeps track of packet loss and ECN marking ratios. It calculates the equivalent delay x_n that accounts for queuing delay, ECN marking, and packet loss, as well as the derivative (i.e., rate of change) of this congestion signal as x'_n . The receiver feeds both pieces of information (x_n and x'_n) back to the sender via periodic RTCP reports.

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

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

4. NADA Receiver Behavior

The receiver continuously monitors end-to-end per-packet statistics in terms of delay, loss, and/or ECN marking ratios. It then aggregates all forms of congestion indicators into the form of an equivalent delay and periodically reports this back to the sender. In addition, the receiver tracks the receiving rate of the flow and includes that in the feedback message.

4.1 Estimation of one-way delay and queuing delay

The delay estimation process in NADA follows a similar approach as in earlier delay-based congestion control schemes, such as LEDBAT [RFC6817]. NADA estimates the forward delay as having a constant base delay component plus a time varying queuing delay component. The base delay is estimated as the minimum value of one-way delay observed over a relatively long period (e.g., tens of minutes), whereas the individual queuing delay value is taken to be the difference between one-way delay and base delay.

In mathematical terms, for packet n arriving at the receiver, one-way delay is calculated as:

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

where $t_{s,n}$ and $t_{r,n}$ are sender and receiver timestamps, respectively. A real-world implementation should also properly handle practical issues such as wrap-around in the value of x_n , which are omitted from the above simple expression for brevity.

The base delay, d_f , is estimated as the minimum value of previously observed x_n 's over a relatively long period. This assumes that the drift between sending and receiving clocks remains bounded by a small value.

Correspondingly, the queuing delay experienced by the packet n is estimated as:

$$d_n = x_n - d_f.$$

The individual sample values of queuing delay should be further filtered against various non-congestion-induced noise, such as spikes due to processing "hiccup" at the network nodes. We denote the resulting queuing delay value as d_{hat_n} .

Our current implementation employs a simple 5-point median filter over per-packet queuing delay estimates, followed by an exponential smoothing filter. We have found such relatively simple treatment to suffice in guarding against processing delay outliers observed in wired connections. For wireless connections with a higher packet delay variation (PDV), more sophisticated techniques on de-noising, outlier rejection, and trend analysis may be needed.

Like other delay-based congestion control schemes, performance of NADA depends on the accuracy of its delay measurement and estimation module. Appendix A in [RFC6817] provides an extensive discussion on this aspect.

4.2 Estimation of packet loss/marketing ratio

The receiver detects packet losses via gaps in the RTP sequence numbers of received packets. It then calculates instantaneous packet loss ratio as the ratio between the number of missing packets over the number of total transmitted packets in the given time window (e.g., during the most recent 500ms). This instantaneous value is passed over an exponential smoothing filter, and the filtered result is reported back to the sender as the observed packet loss ratio p_L .

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

Estimation of packet marking ratio p_M , when ECN is enabled at bottleneck network nodes along the path, will follow the same procedure as above. Here it is assumed that ECN marking information at the IP header are somehow passed along to the transport layer by the receiving endpoint.

4.3 Non-linear warping of delay

In order for a delay-based flow to hold its ground and sustain a reasonable share of bandwidth in the presence of a loss-based flow (e.g., loss-based TCP), it is important to distinguish between different levels of observed queuing delay. For instance, a moderate queuing delay value below 100ms is likely self-inflicted or induced by other delay-based flows, whereas a high queuing delay value of several hundreds of milliseconds may indicate the presence of a loss-based flow that does not refrain from increased delay.

Inspired by the delay-adaptive congestion window backoff policy in [Budzisz-TON11] -- the work by itself is a window-based congestion control scheme with fair coexistence with TCP -- we devise the following non-linear warping of estimated queuing delay value:

$$\begin{aligned} d_{\text{tilde}_n} &= (d_{\text{hat}_n}), & \text{if } d_{\text{hat}_n} < d_{\text{th}}; \\ d_{\text{tilde}_n} &= d_{\text{th}} \frac{(d_{\text{max}} - d_{\text{hat}_n})^4}{(d_{\text{max}} - d_{\text{th}})^4}, & \text{if } d_{\text{th}} < d_{\text{hat}_n} < d_{\text{max}}; \\ d_{\text{tilde}_n} &= 0, & \text{if } d_{\text{hat}_n} > d_{\text{max}}. \end{aligned}$$

Here, the queuing delay value is unchanged when it is below the first threshold d_{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 $(0, d_{\text{th}})$, NADA operates in pure delay-based mode if no losses/markings are present. When queuing delay exceeds d_{max} , NADA reacts to loss/markings 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:

$$x_n = d_{\text{tilde}_n} + p_M d_M + p_L d_L, \quad (1)$$

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

While the value of d_M and d_L are fixed and predetermined in the current design, a scheme for automatically tuning these values based on desired bandwidth sharing behavior in the presence of other competing loss-based flows (e.g., loss-based TCP) is being studied.

In the absence of ECN marking from the network, the value of x_n falls back to the observed queuing delay d_n for packet n when queuing delay is low and no packets are lost over a lightly congested path. In that case the algorithm operates in purely delay-based mode.

4.5 Estimating receiving rate

Estimation of receiving rate of the flow is fairly straightforward. NADA maintains a recent observation window of 500ms, and simply divides the total size of packets arriving during that window over the time span. The receiving rate is denoted as R_r .

4.6 Sending periodic feedback

Periodically, the receiver feeds back a tuple of the most recent values of $\langle d_{\text{hat}_n}, x_n, x'_n, R_r \rangle$ in RTCP feedback messages to aid the sender in its calculation of target rate. The queuing delay value d_{hat_n} is included along with the composite congestion signal x_n so that the sender can decide whether the network is truly underutilized (see Sec. 6.1.1 Accelerated ramp-up).

The value of x'_n corresponds to the derivative (i.e., rate of change) of the composite congestion signal:

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

where the interval between consecutive RTCP feedback messages is denoted as δ . The packet indices corresponding to the current and previous feedback are n and $(n-k)$, respectively.

The choice of target feedback interval needs to strike the right balance between timely feedback and low RTCP feedback message counts. Through simulation studies and frequency-domain analysis, it was determined that a feedback interval below 250ms will not break up the feedback control loop of the NADA congestion control algorithm. Thus, it is recommended to use a target feedback interval of 100ms. This will result in a feedback bandwidth of 16Kbps with 200 bytes per feedback message, less than 0.1% overhead for a 1Mbps flow.

4.7 Discussions on delay metrics

The current design works with relative one-way-delay (OWD) as the main indication of congestion. The value of the relative OWD is obtained by maintaining the minimum value of observed OWD over a relatively long time horizon and subtract that out from the observed absolute OWD value. Such an approach cancels out the fixed difference between the sender and receiver clocks. It has been widely adopted by other delay-based congestion control approaches such as LEDBAT [RFC6817]. As discussed in [RFC6817], the time horizon for tracking the minimum OWD needs to be chosen with care: it must be long enough for an opportunity to observe the minimum OWD with zero queuing delay along the path, and sufficiently short so as to timely reflect "true" changes in minimum OWD introduced by route changes and other rare events.

The potential drawback in relying on relative OWD as the congestion signal is that when multiple flows share the same bottleneck, the flow arriving late at the network experiencing a non-empty queue may mistakenly consider the standing queuing delay as part of the fixed path propagation delay. This will lead to slightly unfair bandwidth sharing among the flows.

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

Note that the choice of either delay metric (relative OWD vs. RTT) involves no change in the proposed rate adaptation algorithm at the sender. Therefore, comparing the pros and cons regarding which delay metric to adopt can be kept as an orthogonal direction of investigation.

5. NADA Sender Behavior

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

In the following, we describe these modules in further detail, and explain how they interact with each other.

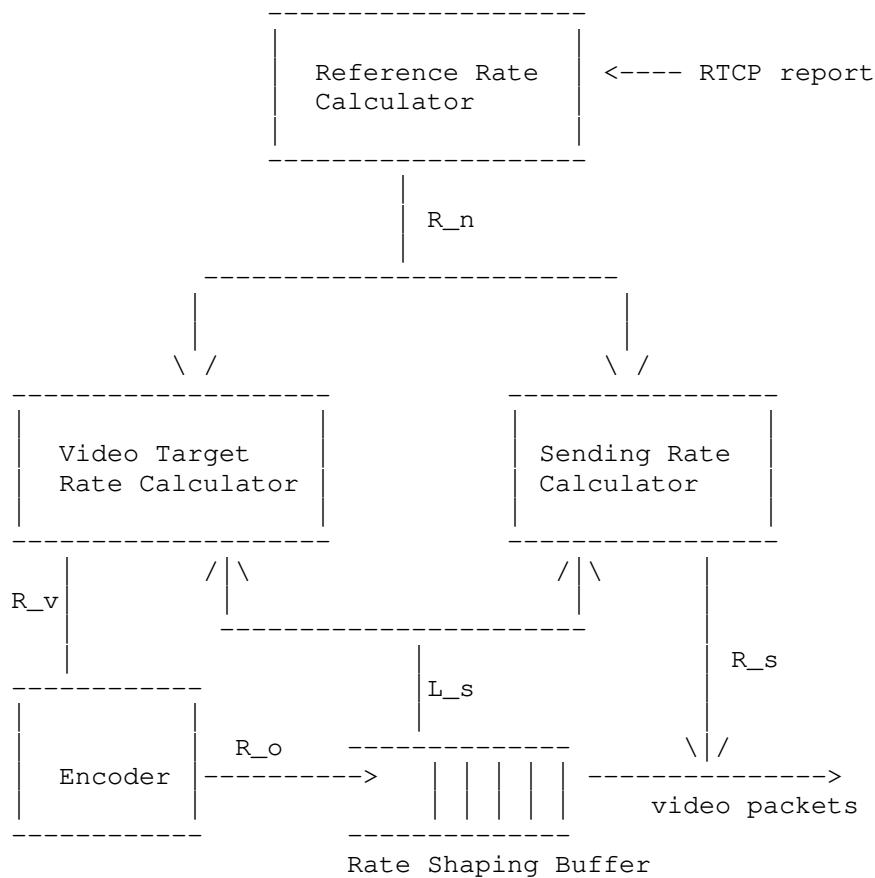


Figure 1 NADA Sender Structure

5.1 Reference rate calculation

The sender initializes the reference rate R_n as R_{\min} by default, or to a value specified by the upper-layer application. [Editor's note: should proper choice of starting rate value be within the scope of the CC solution?]

The reference rate R_n is calculated based on receiver feedback information regarding queuing delay d_{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:

$$\gamma = \min \left[\gamma_0, \frac{T_{\text{th}}}{\text{RTT}_0 + \delta_0} \right] \quad (3)$$

$$R_n = (1 + \gamma) R_r \quad (4)$$

In (3) and (4), the multiplier γ for rate increase is upper-bounded by a fixed ratio γ_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 δ_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), δ_s refers to the time interval between current and previous rate updates. Note that δ_s is the same as the RTCP report interval at the receiver (see δ from (2)) when the backward path is uncongested.

In (6), R_{min} and R_{max} denote the content-dependent rate range the encoder can produce. The weighting factor reflecting a flow's priority is w . The reference congestion signal x_{ref} is chosen so that the maximum rate of R_{max} can be achieved when $x_{hat} = w * x_{ref}$.

Proper choice of the scaling parameters η and κ in (5) and (7) can ensure system stability so long as the RTT falls below the upper bound of τ_o . The recommended default value of τ_o is chosen as 500ms.

For both modes of operations, the final reference rate R_n is clipped within the range of $[R_{min}, R_{max}]$. Note also that the sender does not need any explicit knowledge of the management scheme inside the network. Rather, it reacts to the aggregation of all forms of congestion indications (delay, loss, and explicit markings) via the composite congestion signals x_n and x'_n from the receiver in a coherent manner.

5.2 Video encoder rate control

The video encoder rate control procedure has the following characteristics:

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

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

5.3 Rate shaping buffer

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

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

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

5.4 Adjusting video target rate and sending rate

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

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

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

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

In (8) and (9), the first term indicates the rate calculated from network congestion feedback alone. The second term indicates the influence of the rate shaping buffer. A large rate shaping buffer nudges the encoder target rate slightly below -- and the sending rate slightly above -- the reference rate R_n .

Intuitively, the amount of extra rate offset needed to completely drain the rate shaping buffer within the same time frame of encoder rate adaptation τ_v is given by L_s/τ_v . The scaling parameters β_v and β_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 Evaluation Results", presented at IETF 90, <https://tools.ietf.org/agenda/90/slides/slides-90-rmcat-6.pdf>
- [IETF-91] Zhu, X. et al., "NADA Algorithm Update and Test Case Evaluations", presented at IETF 91 Interim, <https://datatracker.ietf.org/meeting/91/agenda/rmcat/>

Appendix A. Network Node Operations

NADA can work with different network queue management schemes and does not assume any specific network node operation. As an example, this appendix describes three variations of queue management behavior at the network node, leading to either implicit or explicit congestion signals.

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

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

A.1 Default behavior of drop tail

In a conventional network with drop tail or RED queues, congestion is inferred from the estimation of end-to-end delay and/or packet loss. Packet drops at the queue are detected at the receiver, and contributes to the calculation of the equivalent delay x_n . No special action is required at network node.

A.2 ECN marking

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

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

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

The smoothing parameter α is a value between 0 and 1. A value of $\alpha=1$ corresponds to performing no smoothing at all.

* calculate marking probability p as:

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

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

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

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

The ECN markings events will contribute to the calculation of an equivalent delay x_n at the receiver. No changes are required at the sender.

A.3 PCN marking

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


```
* upon packet arrival, meter packet against token bucket (r,b);  
* update token level b_tk;  
* calculate the marking probability as:  
  
  p = 0, if b-b_tk < b_lo;  
  
  p = p_max *  $\frac{b-b_{tk}-b_{lo}}{b_{hi}-b_{lo}}$ , if b_lo <= b-b_tk < b_hi;  
  
  p = 1, if b-b_tk >= b_hi.
```

Here, the token bucket lower and upper limits are denoted by b_{lo} and b_{hi} , respectively. The parameter b indicates the size of the token bucket. The parameter r is chosen as $r=\gamma*C$, 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