Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

M. Zanaty
Cisco
V. Singh
Aalto University
S. Nandakumar
Cisco
Z. Sarkar
Ericsson AB

March 18, 2016

Congestion Control and Codec interactions in RTP Applications
draft-ietf-rmcat-cc-codec-interactions-02

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 19, 2016.

Copyright Notice

Table of Contents

1. Introduction

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

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

- Some applications have no explicit congestion control, despite the clear requirements in RTP and its profiles AVP [RFC3551] and AVPF [RFC4585], under the expectation that users will terminate media flows that are significantly impaired by congestion (in essence, human circuit breakers).

- For media with substantially higher packet rates and bit rates, such as many video codecs, various non-standard congestion control techniques are often used to adapt transmission rate based on receiver feedback.

- Some experimental applications use standardized techniques such as TCP-Friendly Rate Control (TFRC) [RFC5348]. However, for various reasons, these have not been widely deployed.

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

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

In order to meet these requirements, interactions are necessary between the application’s congestion controller, the RTP layer, media codecs, other components, and the underlying UDP/IP network stack. This memo attempts to present a conceptual model of the various interfaces based on a simplified application decomposition. This memo discusses interactions between the congestion control and codec control layer in a typical RTP Application.

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

2. Key Words for Requirements

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

3. Conceptual Model

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

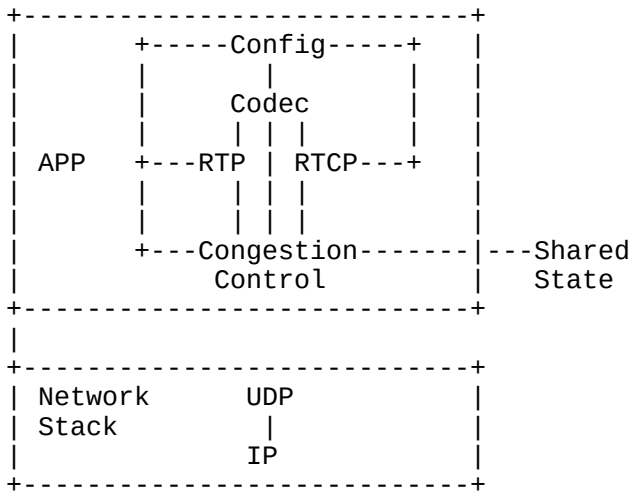


Figure 1

- APP: Application containing one or more RTP streams and the corresponding media codecs and congestion controllers. For example, a WebRTC browser.

- Config: Configuration specified by the application that provides the media and transport parameters, RTP and RTCP parameters and extensions, and congestion control parameters. For example, a WebRTC Javascript application may use the 'constraints' API to affect the media configuration, and SDP applications may negotiate the media and transport parameters with the remote peer. This determines the initial static configuration negotiated in session establishment. The dynamic state may differ due to congestion or other factors, but still must conform to limits established in the config.

- Codec: Media encoder/decoder or other source/sink for the RTP payload. The codec may be, for example, a simple monaural audio format, a complex scalable video codec with several dependent

layers, or a source/sink with no live encoding/decoding such as a mixer which selectively switches and forwards streams rather than mixes media.

- RTP: Standard RTP stack functions, including media packetization / de-packetization and header processing, but excluding existing extensions and possible new extensions specific to congestion control (CC) such as absolute timestamps or relative transmission time offsets in RTP header extensions. RTCP: Standard RTCP functions, including sender reports, receiver reports, extended reports, circuit breakers [I-D.ietf-avtcore-rtp-circuit-breakers], feedback messages such as NACK [RFC4585] and codec control messages such as TMMBR [RFC5104], but excluding existing extensions and possible new extensions specific to congestion control (CC) such as REMB [I-D.alvestrand-rmcat-remb] (for receiver-side CC), ACK (for sender-side CC), absolute and/or relative timestamps (for sender-side or receiver-side CC), etc.

- Congestion Control: All functions directly responsible for congestion control, including possible new RTP/RTCP extensions, send rate computation (for sender-side CC), receive rate computation (for receiver-side CC), other statistics, and control of the UDP sockets including packet scheduling for traffic shaping/pacing.

- Shared State: Storage and exchange of congestion control state for multiple flows within the application and beyond it.

- Network Stack: The platform’s underlying network functions, usually part of the Operating System (OS), containing the UDP socket interface and other network functions such as ECN, DSCP, physical interface events, interface-level traffic shaping and packet scheduling, etc. This is usually part of the Operating System, often within the kernel; however, user-space network stacks and components are also possible.

4. Implementation Model

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

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

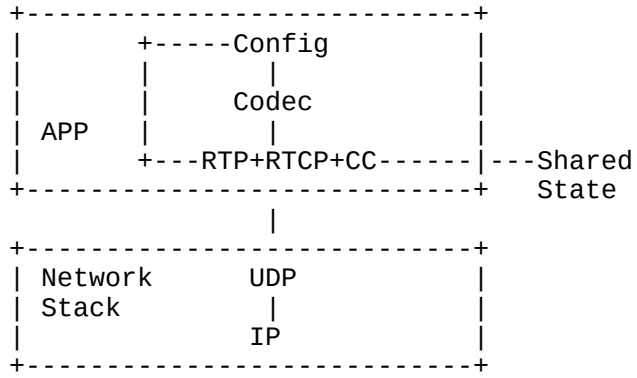


Figure 2

5. Codec - CC Interactions

The following subsections identify the necessary interactions between the Codec and congestion control (CC) layer interfaces that needs to be considered important.

5.1. Mandatory Interactions

5.1.1. Allowed Rate

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

5.2. Optional Interactions

This section identifies certain advanced interactions that if implemented by an RMCAT solution shall provide more granular control over the congestion control state and the encoder behavior. As of today, these interactions are optional to implement and future evaluations of the existing/upcoming codecs might result in considering some or all of these as Mandatory interactions.

5.2.1. Media Elasticity

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

5.2.2. Startup Ramp

Startup Ramp (from Codec to CC, and from CC to Codec): Startup is an important moment in a conversation. Rapid rate adaptation during startup is therefore important. The codec should minimize its startup media rate as much as possible without adversely impacting the user experience, and support a strategy for rapid rate ramp. The CC should allow the highest startup media rate as possible without adversely impacting network conditions, and also support rapid rate ramp until stabilizing on the available bandwidth. Startup can be viewed as a negotiation between the codec and the CC. The specification of the ramp may take a number of forms depending on the interface to the codec; for example, a percentage bit rate increase per RTT (or other time interval), or an increased transmit window (in number of packets and/or octets allowed outstanding) are all potential forms. The codec requests a startup rate and ramp, and the CC responds with the allowable parameters which may be lower/slower. The RMCAT requirements also include the possibility of bandwidth history to further accelerate or even eliminate startup ramp time. While this acceleration or elimination in ramp time is beneficial to the session user experience when bandwidth is sufficient, it can be detrimental if significant congestion results (the user experience of this session and all other sessions traversing the point of

congestion may unnecessarily degrade). Therefore, it is recommended that use of potentially stale congestion state for acceleration or elimination in ramp up be limited to topologies or deployments believed to have sufficient bandwidth margin or those in which the potential transient congestion risk is acceptable. Note that startup can often commence before user interaction or conversation to reduce the chance of clipped media.

5.2.3. Delay Tolerance

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

5.2.4. Loss Tolerance

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

5.2.5. Forward Error Correction

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

5.2.6. Probing for Available Bandwidth

FEC can also be used to probe for additional available bandwidth, if the application desires a higher target rate than the current rate. FEC is preferable to synthetic probes since any contribution to congestion by the FEC probe will not impact the post-repair loss rate of the source media flow while synthetic probes may adversely affect the loss rate. Note that any use of FEC or retransmission must ensure that the total flow of all packets including FEC, retransmission and original media never exceeds the Allowed Rate.

5.2.7. Throughput Sensitivity

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

5.2.8. Rate Stability

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

6. Acknowledgements

The RMCAT design team discussions contributed to this memo. The authors would also like to thank Michael Ramalho for reviewing and suggesting text improvements.

7. IANA Considerations

This memo includes no request to IANA.

8. References

8.1. Normative References

[I-D.alvestrand-rmcat-remb]
Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", draft-alvestrand-rmcat-remb-03 (work in progress), October 2013.

[I-D.ietf-avtcore-rtp-circuit-breakers]
Perkins, C. and V. Varun, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", draft-ietf-avtcore-rtp-circuit-breakers-14 (work in progress), March 2016.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-27 (work in progress), February 2016.

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

[I-D.welzl-rmcat-coupled-cc]
Welzl, M., Islam, S., and S. Gjessing, "Coupled congestion control for RTP media", draft-welzl-rmcat-coupled-cc-05 (work in progress), June 2015.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

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

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

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <http://www.rfc-editor.org/info/rfc3551>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <http://www.rfc-editor.org/info/rfc4340>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <http://www.rfc-editor.org/info/rfc5104>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <http://www.rfc-editor.org/info/rfc5109>.

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

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, DOI 10.17487/RFC5450, March 2009, <http://www.rfc-editor.org/info/rfc5450>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <http://www.rfc-editor.org/info/rfc5506>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, DOI 10.17487/RFC5761, April 2010, <http://www.rfc-editor.org/info/rfc5761>.

[RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, DOI 10.17487/RFC6330, August 2011, <http://www.rfc-editor.org/info/rfc6330>.

[RFC6865] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6865, DOI 10.17487/RFC6865, February 2013, <http://www.rfc-editor.org/info/rfc6865>.

8.2. Informative References

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

Authors' Addresses

Mo Zanaty
Cisco

Email: mzanaty@cisco.com

Varun Singh
Aalto University

Email: varun@comnet.tkk.fi

Suhas Nandakumar
Cisco

Email: snandaku@cisco.com

Zaheduzzaman Sarker
Ericsson AB

Email: zaheduzzaman.sarker@ericsson.com

RTP Media Congestion Avoidance Techniques (rmcat)
Internet-Draft
Intended status: Experimental
Expires: October 16, 2016

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

April 14, 2016

Coupled congestion control for RTP media
draft-ietf-rmcat-coupled-cc-02

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 both the NADA and Google congestion control algorithms.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 16, 2016.

Copyright Notice

Table of Contents

1. Introduction

When there is enough data to send, a congestion controller must increase its sending rate until the path's capacity has been reached; depending on the controller, sometimes the rate is increased further,

until packets are ECN-marked or dropped. This process inevitably creates undesirable queuing delay -- an effect that is amplified when multiple congestion controlled connections traverse the same network bottleneck.

The Congestion Manager (CM) [RFC3124] couples flows by providing a single congestion controller. 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 to combine the behavior of congestion control mechanisms that is easier to implement than the Congestion Manager [RFC3124] 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. It is able to honor user-specified priorities, which is required by rtcweb [RFC7478].

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). This document describes both active and passive versions, however the passive version is put into the appendix as it is extremely experimental.

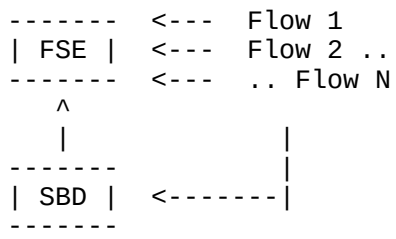


Figure 1: Coupled congestion control architecture

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]). Such multiplexing is also a recommended usage of RTP in 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 [I-D.ietf-rmcat-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 [I-D.ietf-rmcat-sbd] for details).

5.2. FSE

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

- a unique flow number to identify the flow
- the FGI of the FG that it belongs to (based on the definitions in this document, a flow has only one bottleneck, and can therefore be in only one FG)
- a priority P, which here is assumed to be represented as a floating point number in the range from 0.1 (unimportant) to 1 (very important).
- The rate used by the flow in bits per second, FSE_R.

Note that the priority does not need to be a floating point value and its value range does not matter for this algorithm: the algorithm works with a flow's priority portion of the sum of all priority values. Priorities can therefore be mapped to the "very-low", "low", "medium" or "high" priority levels described in [I-D.ietf-rtcweb-transports] using the values 1, 2, 4 and 8, respectively.

The FSE can operate on window-based as well as rate-based congestion controllers. 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 the sum of the calculated rates of all flows in the same FG. This value is used to calculate the sending rate.

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

5.3. Flows

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

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

5.3.1. Example algorithm 1 - Active FSE

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

(1) When a flow f starts, it registers itself with SBD and the FSE. FSE_R is initialized with the congestion controller's initial rate. SBD will assign the correct FGI. When a flow is assigned an FGI, it adds its FSE_R to S_CR.

(2) When a flow f stops, its entry is removed from the list.

(3) Every time the congestion controller of the flow f determines a new sending rate CC_R, the flow calls UPDATE, which carries out the tasks listed below to derive the new sending rates for all the flows in the FG. A flow's UPDATE function uses a local (i.e. per-flow) temporary variable S_P, which is the sum of all the priorities.

(a) It updates S_CR.

```
S_CR = S_CR + CC_R - FSE_R(f)
```

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

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

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

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

5.3.2. Example algorithm 2 - Conservative Active FSE

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

(1) When a flow f starts, it registers itself with SBD and the FSE. FSE_R is initialized with the congestion controller's initial rate. SBD will assign the correct FGI. When a flow is assigned an FGI, it adds its FSE_R to S_CR.

(2) When a flow f stops, its entry is removed from the list.

(3) Every time the congestion controller of the flow f determines a new sending rate CC_R, the flow calls UPDATE, which carries out the tasks listed below to derive the new sending rates for all

the flows in the FG. A flow's UPDATE function uses a local (i.e. per-flow) temporary variable S_P, which is the sum of all the priorities, and a local variable DELTA, which is used to calculate the difference between CC_R and the previously stored FSE_R. To prevent flows from either ignoring congestion or overreacting, a timer keeps them from changing their rates immediately after the common rate reduction that follows a congestion event. This timer is set to 2 RTTs of the flow that experienced congestion because it is assumed that a congestion event can persist for up to one RTT of that flow, with another RTT added to compensate for fluctuations in the measured RTT value.

(a) It updates S_CR based on DELTA.

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

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

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

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

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

6. Application

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

6.1. NADA

Network-Assisted Dynamic Adapation (NADA) [I-D.ietf-rmcat-nada] is a congestion control scheme for rtcweb. It calculates a reference rate r_ref upon receiving an acknowledgment, and then, based on the reference rate, it calculates a video target rate r_vin and a sending rate for the flows, r_send.

When applying the FSE to NADA, the UPDATE function call described in Section 5.3 gives the FSE NADA's reference rate r_ref. 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_ref(r_vin and r_send) of flow i with the value of FSE_R(i).

6.2. GCC

Google Congestion Control (GCC) [I-D.ietf-rmcat-gcc] is another congestion control scheme for rtcweb. The rate control of GCC employs two parts: controlling the bandwidth estimate based on delay, and controlling the bandwidth estimate based on loss. Both are designed to estimate the available bandwidth, A_hat.

When applying the FSE to GCC, the UPDATE function call described in Section 5.3 gives the FSE GCC's estimate of available bandwidth A_hat. The recommended algorithm for GCC 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 A_hat of flow i with the value of FSE_R(i).

6.3. General recommendations

This section provides general advice for applying the FSE to congestion control mechanisms.

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, Karen Nielsen, Andreas Petlund, David Ros (who also gave the FSE its name), Zaheduzzaman Sarker, Varun

Singh and Kristian Hiorth. The authors would like to thank Xiaoqing Zhu and Stefan Holmer for helping with NADA and GCC.

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

[I-D.ietf-rmcat-gcc]
Holmer, S., Lundin, H., Carlucci, G., Cicco, L., and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication", draft-ietf-rmcat-gcc-01 (work in progress), October 2015.

[I-D.ietf-rmcat-nada]
Zhu, X., Pan, R., Ramalho, M., Cruz, S., Jones, P., Fu, J., D'Aronco, S., and C. Ganzhorn, "NADA: A Unified Congestion Control Scheme for Real-Time Media", draft-ietf-rmcat-nada-02 (work in progress), March 2016.

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

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, DOI 10.17487/RFC3124, June 2001, <http://www.rfc-editor.org/info/rfc3124>.

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

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) and ACM SIGCOMM CCR 44(4) 2014; extended version available as a technical report from http://safiquli.at.ifi.uio.no/paper/fse-tech-report.pdf , 2014.

[fse-noms]
Islam, S., Welzl, M., Hayes, D., and S. Gjessing, "Managing Real-Time Media Flows through a Flow State Exchange", IEEE NOMS 2016, Istanbul, Turkey , 2016.

[I-D.ietf-rmcat-sbd]
Hayes, D., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.", draft-ietf-rmcat-sbd-04 (work in progress), March 2016.

[I-D.ietf-rtcweb-transports]
Alvestrand, H., "Transports for WebRTC", Internet-draft draft-ietf-rtcweb-transports-11.txt, January 2016.

[RFC7478] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", RFC 7478, DOI 10.17487/RFC7478, March 2015, <http://www.rfc-editor.org/info/rfc7478>.

[rtcweb-rtp-usage]
Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", Internet-draft draft-ietf-rtcweb-rtp-usage-26.txt, March 2016.

[transport-multiplex]
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a Single Lower-Layer Transport", Internet-draft draft-westerlund-avtcore-transport-multiplexing-07.txt, October 2013.

Appendix A. Scheduling

When connections originate from the same host, it would be possible 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 as a function of the scheduler. The Congestion Manager (CM) [RFC3124] also uses such a scheduling function.

Appendix B. 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. Results of a simplified passive FSE algorithm with both NADA and GCC can be found in [fse-noms].

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

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

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

(1) When a flow f starts, it registers itself with SBD and the FSE. FSE_R and DR are initialized with the congestion controller's initial rate. SBD will assign the correct FGI. When a flow is assigned an FGI, it adds its FSE_R to S_CR.

(2) When a flow f stops, it sets its DR to 0 and sets P to -1.

(3) Every time the congestion controller of the flow f determines a new sending rate CC_R, assuming the flow's new desired rate new_DR to be "infinity" in case of a bulk data transfer with an unknown maximum rate, the flow calls UPDATE, which carries out the tasks listed below to derive the flow's new sending rate, Rate. A flow's UPDATE function uses a few local (i.e. per-flow) temporary variables, which are all initialized to 0: DELTA, new_S_CR and S_P.

    (a) For all the flows in its FG (including itself), it calculates the sum of all the calculated rates, new_S_CR. Then it calculates the difference between FSE_R(f) and CC_R, DELTA.

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

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

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

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

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

(d) It calculates the sending rate, Rate.

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

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

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

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

The goals of the flow algorithm are to achieve prioritization, improve network utilization in the face of application-limited flows, and impose limits on the increase behavior such that the negative impact of multiple flows trying to increase their rate together is minimized. It does that by assigning a flow a sending rate that may not be what the flow's congestion controller expected. It therefore builds on the assumption that no significant inefficiencies arise

from temporary application-limited behavior or from quickly jumping to a rate that is higher than the congestion controller intended. How problematic these issues really are depends on the controllers in use and requires careful per-controller experimentation. The coupled congestion control mechanism described here also does not require all controllers to be equal; effects of heterogeneous controllers, or homogeneous controllers being in different states, are also subject to experimentation.

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

B.1. Example operation (passive)

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

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

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

S_CR = 1, TLO = 0

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

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

S_CR = 10, TLO = 0

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

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

S_CR = 11, TLO = 0

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

CC_R = 8. new_DR = infinity.
3 a) new_S_CR = 11; DELTA = 8 - 10 = -2.
3 b) FSE_Rf) = 8. DELTA is negative, hence S_CR = 9;
DR(f) = 8.
3 c) S_P = 1.5.
3 d) new sending rate = min(infinity, 1/1.5 * 9 + 0) = 6.
3 e) FSE_R(f) = 6.

The resulting FSE looks as follows:

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

S_CR = 9, TLO = 0

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

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

The resulting FSE looks as follows:

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

S_CR = 10, TLO = 0

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

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

The resulting FSE looks as follows:

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

S_CR = 11, TLO = 5.33

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

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

The resulting FSE looks as follows:

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

S_CR = 12, TLO = 0

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

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

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

The resulting FSE looks as follows:

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

S_CR = 9.33, TLO = 0

Appendix C. Change log

C.1. draft-welzl-rmcat-coupled-cc

C.1.1. Changes from -00 to -01

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

C.1.2. Changes from -01 to -02

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

C.1.3. Changes from -02 to -03

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

- Moved the passive variant of the algorithm to appendix.

C.1.4. Changes from -03 to -04

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

C.1.5. Changes from -04 to -05

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

C.2. draft-ietf-rmcat-coupled-cc

C.2.1. Changes from draft-welzl-rmcat-coupled-cc-05

- Moved scheduling section to the appendix.

C.2.2. Changes from -00 to -01

- Included how to apply the algorithm to GCC.
- Updated variable names of NADA to be in line with the latest version.
- Added a reference to [I-D.ietf-rtcweb-transports] to make a connection to the prioritization text there.

C.2.3. Changes from -01 to -02

- Minor changes.
- Moved references of NADA and GCC from informative to normative.
- Added a reference for the passive variant of the algorithm.

Authors' Addresses

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

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

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

Phone: +47 22 85 24 20
Email: michawe@ifi.uio.no

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

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

RMCAT WG
Internet-Draft
Intended status: Informational
Expires: September 22, 2016

V. Singh
callstats.io
J. Ott
Technical University of Munich
S. Holmer
Google

March 21, 2016

Evaluating Congestion Control for Interactive Real-time Media
draft-ietf-rmcat-eval-criteria-05

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.

Table of Contents

1. Introduction

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

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

This document only provides broad-level criteria for evaluating a new congestion control algorithm. The minimal requirement for RMCAT proposals is to produce or present results for the test scenarios described in [I-D.ietf-rmcat-eval-test] (Basic Test Cases). Additionally, proponents may produce evaluation results for the wireless test scenarios [I-D.ietf-rmcat-wireless-tests].

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

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 (measured at 200ms intervals)

2. Packets sent, Packets received

3. Bytes sent, bytes received

4. Packet delay

5. Packets lost, Packets discarded (from the playout or de-jitter buffer)

6. If using, retransmission or FEC: post-repair loss

7. Fairness or Unfairness: Experiments testing the performance of an RMCAT proposal against any cross-traffic must define its expected criteria for fairness. The "unfairness" test guideline (measured at 1s intervals) is:
   1. Does not trigger the circuit breaker.
   2. No RMCAT stream achieves more than 3 times the average throughput of the RMCAT stream with the lowest average throughput, for a case when the competing streams have similar RTTs.
   3. RTT should not grow by a factor of 3 for the existing flows when a new flow is added.
   For example, see the test scenarios described in [I-D.ietf-rmcat-eval-test].

8. 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.

9. Instability or oscillation in the sending rate: The frequency or number of instances when the sending rate oscillates between an high watermark level and a low watermark level, or vice-versa in a defined time window. For example, the watermarks can be set at 4x interval: 500 Kbps, 2 Mbps, and a time window of 500ms.

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

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

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

3.1. RTP Log Format

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

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

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

4. List of Network Parameters

The implementors initially are encouraged to choose evaluation settings from the following values:

4.1. One-way Propagation Delay

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

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

4.2. End-to-end Loss

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

1. no loss: 0%
2. 1%
3. 5%

4. 10%

5. 20%

4.3. DropTail Router Queue Length

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

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

2. Nominal: 300-500ms

3. Buffer-bloated: 1000-2000ms

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

QueueSize (in bytes) = QueueSize (in sec) x Throughput (in bps)/8

4.4. Loss generation model

[Open Issue: Describes the model for generating packet losses, for example, losses can be generated using traces, or using the Gilbert-Elliot model, or randomly (uncorrelated loss).]

4.5. Jitter models

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

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

Because jitter is an overloaded term, we instead use the term Packet Delay Variation (PDV) to describe the variation of delay of

individual packets in the same sense as the IETF IPPM WG has defined PDV in their documents (e.g., RFC 3393) and as the ITU-T SG16 has defined IP Packet Delay Variation (IPDV) in their documents (e.g., Y.1540).

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

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

[Open issue: which one is used in evaluations? Are both used?]

4.5.1. Random Bounded PDV (RBPDV)

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

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

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

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

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

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

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

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

1 - Two-sided: Uniform PDV Distribution. Two quantities to define: x_min and x_max.

2 - Two-sided: Truncated Gaussian PDV Distribution. Four quantities to define: the appropriate x_min and x_max for test (e.g., +/- two sigma values), the standard deviation, and the mean.

3 - One Sided: Truncated Gaussian PDV Distribution. Quantities to define: three sigma value, the standard deviation, and the mean]

5. WiFi or Cellular Links

[I-D.ietf-rmcat-wireless-tests] describes the test cases to simulate networks with wireless links. The document describes mechanism to simulate both cellular and WiFi networks.

6. Traffic Models

6.1. TCP taffic model

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

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

The short TCP flows can be modelled as follows: 30 connections start simultaneously fetching small (30-50 KB) amounts of data. This covers the case where the short TCP flows are not fetching a video file.

The idle period between bursts of starting a group of TCP flows is typically derived from an exponential distribution with the mean value of 10 seconds.

[These values were picked based on the data available at http://httparchive.org/interesting.php as of October 2015].

6.2. RTP Video model

[I-D.ietf-rmcat-video-traffic-model] describes two types of video traffic models for evaluating RMCAT candidate algorithms. The first model statistically characterizes the behavior of a video encoder. Whereas the second model uses video traces.

For example, test sequences are available at: [xiph-seq] and [HEVC-seq].

[Open issue: Which sequences are used? All? Some subset?]

6.3. Background UDP

[Open issue: Background UDP flow is modeled as a constant bit rate (CBR) flow. It will download data at a particular CBR rate for the complete session, or will change to particular CBR rate at predefined intervals. They parameters are still TBD. e.g., packet size, packet spacing interval, etc.]

7. Security Considerations

Security issues have not been discussed in this memo.

8. IANA Considerations

There are no IANA impacts in this memo.

9. Contributors

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

Michael Ramalho provided the text for the Jitter model.

10. Acknowledgements

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

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

11. References

11.1. Normative References

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

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <http://www.rfc-editor.org/info/rfc3551>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <http://www.rfc-editor.org/info/rfc5506>.

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

[I-D.ietf-avtcore-rtp-circuit-breakers] Perkins, C. and V. Varun, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", draft-ietf-avtcore-rtp-circuit-breakers-14 (work in progress), March 2016.

[I-D.ietf-rmcat-wireless-tests] Sarker, Z., Johansson, I., Zhu, X., Fu, J., Tan, W., and M. Ramalho, "Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks", draft-ietf-rmcat-wireless-tests-01 (work in progress), November 2015.

11.2. Informative References

[RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, DOI 10.17487/RFC5033, August 2007, <http://www.rfc-editor.org/info/rfc5033>.

[RFC5166] Floyd, S., Ed., "Metrics for the Evaluation of Congestion Control Mechanisms", RFC 5166, DOI 10.17487/RFC5166, March 2008, <http://www.rfc-editor.org/info/rfc5166>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[I-D.ietf-rmcat-eval-test]
Sarker, Z., Varun, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-ietf-rmcat-eval-test-03 (work in progress), March 2016.

[I-D.ietf-rmcat-video-traffic-model]
Zhu, X., Cruz, S., and Z. Sarker, "Modeling Video Traffic Sources for RMCAT Evaluations", draft-ietf-rmcat-video-traffic-model-00 (work in progress), January 2016.

[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.

[xiph-seq]
Xiph.org, , "Video Test Media", http://media.xiph.org/video/derf/ , .

[HEVC-seq]
HEVC, , "Test Sequences", http://www.netlab.tkk.fi/~varun/test_sequences/ , .

Appendix A. Application Trade-off

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

A.1. Measuring Quality

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

Appendix B. Change Log

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

B.1. Changes in draft-ietf-rmcat-eval-criteria-05

- Improved text surrounding wireless tests, video sequences, and short-TCP model.

B.2. Changes in draft-ietf-rmcat-eval-criteria-04

- Removed the guidelines section, as most of the sections are now covered: wireless tests, video model, etc.
- Improved Short TCP model based on the suggestion to use httparchive.org.

B.3. Changes in draft-ietf-rmcat-eval-criteria-03

- Keep-alive version.
- Moved link parameters and traffic models from eval-test

B.4. Changes in draft-ietf-rmcat-eval-criteria-02

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

B.5. Changes in draft-ietf-rmcat-eval-criteria-01

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

B.6. Changes in draft-ietf-rmcat-eval-criteria-00

- Updated references.
- Resubmitted as WG draft.

B.7. Changes in draft-singh-rmcat-cc-eval-04

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

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

B.8. Changes in draft-singh-rmcat-cc-eval-03

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

B.9. Changes in draft-singh-rmcat-cc-eval-02

- Added scenario descriptions.

B.10. Changes in draft-singh-rmcat-cc-eval-01

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

Authors' Addresses

Varun Singh
Nemu Dialogue Systems Oy
Runeberginkatu 4c A 4
Helsinki 00100
Finland

Email: varun.singh@iki.fi
URI: http://www.callstats.io/

Joerg Ott
Technical University of Munich
Faculty of Informatics
Boltzmannstrasse 3
Garching bei Muenchen, DE 85748
Germany

Email: ott@in.tum.de

Stefan Holmer
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: holmer@google.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 9, 2016

Z. Sarker
Ericsson AB
V. Singh
callstats.io
X. Zhu
M. Ramalho
Cisco Systems
March 08, 2016

Test Cases for Evaluating RMCAT Proposals
draft-ietf-rmcat-eval-test-03

Abstract

The Real-time Transport Protocol (RTP) is used to transmit media in multimedia telephony applications, these applications are typically required to implement congestion control. This document describes the test cases to be used in the performance evaluation of such congestion control algorithms.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 9, 2016.

Table of Contents

1. Introduction

This memo describes a set of test cases for evaluating congestion control algorithm proposals for real-time interactive media. It is based on the guidelines enumerated in [I-D.ietf-rmcat-eval-criteria] and the requirements discussed in [I-D.ietf-rmcat-cc-requirements]. The test cases cover basic usage scenarios and are described using a common structure, which allows for additional test cases to be added to those described herein to accommodate other topologies and/or the modeling of different path characteristics. The described test cases

in this memo SHOULD be used to evaluate any proposed congestion control algorithm for real-time interactive media.

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. Structure of Test cases

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

- Define the test case:
  - General description: describes the motivation and the goals of the test case.
  - Expected behavior: describes the desired rate adaptation behavior.
  - Define a list of metrics to evaluate the desired behavior: this indicates the minimum set of metrics (e.g., link utilization, media sending rate) that a proposed algorithm needs to measure to validate the expected rate adaptation behavior. It should also indicate the time granularity (e.g., averaged over 10ms, 100ms, or 1s) for measuring certain metrics. Typical measurement interval is 200ms.
- Define testbed topology: every test case needs to define an evaluation testbed topology. Figure 1 shows such an evaluation topology. In this evaluation topology, S1..Sn are traffic sources. These sources generate media traffic and use either congestion control algorithm under investigation. R1..Rn are the corresponding receivers. A test case can have one or more such traffic sources (S) and their corresponding receivers (R). The path from the source to destination is denoted as "forward" and the path from a destination to a source is denoted as "backward". The following basic structure of the test case has been described from the perspective of media generating endpoints attached on the

left-hand side of Figure 1. In this setup, the media flows are transported in forward direction and corresponding feedback/ control messages are transported in the backward direction. However, it is also possible to set up the test with media in both forward and backward directions. In that case, unless otherwise specified by the test case, it is expected that the backward path does not introduce any congestion related impairments and has enough capacity to accommodate both media and feedback/control messages. It should be noted that depending on the test cases it is possible to have different path characteristics in either of the directions.

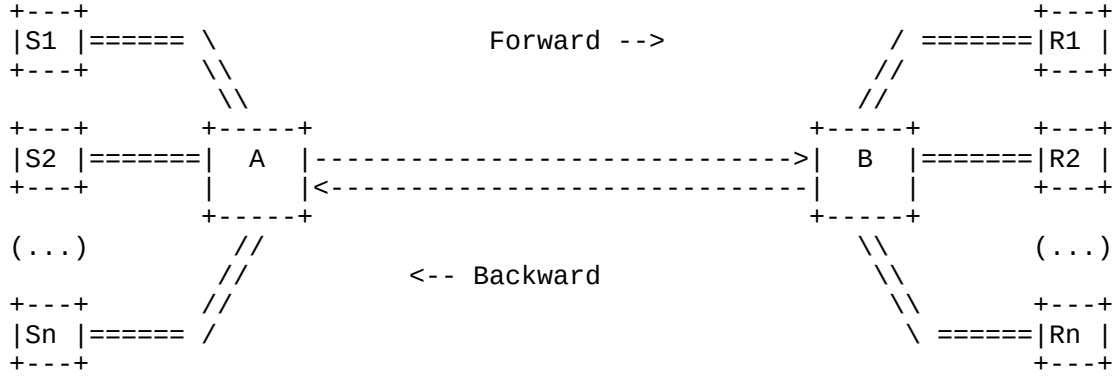


Figure 1: Example of A Testbed Topology

In a testbed environment where real equipments are used to create a laboratory, there may exist a significant amount of traffic on portions of the network path between the endpoints that is not desired for the purposes of the tests described in the document. Some of this traffic may be generated by other processes on the endpoints themselves (e.g., discovery protocols) or by other endpoints not presently under test. It is recommended not to route traffic generated by endpoints that are not under test through the test bed and route those traffic generated by the endpoints under test around the bottleneck links specified herein.

- Define testbed attributes:
  - Duration: defines the duration of the test in seconds.
  - Path characteristics: defines the end-to-end transport level path characteristics of the testbed for a particular test case. Two sets of attributes describe the path characteristics, one for the forward path and the other for the backward path. The path characteristics for a particular path direction is

applicable to all the Sources "S" sending traffic on that path. If only one attribute is specified, it is used for both path directions, however, unless specified the reverse path has no capacity restrictions and no path loss.

+ Path direction: forward or backward.

+ Bottleneck-link capacity: defines minimum capacity of the end-to-end path

+ Reference bottleneck capacity: defines a reference value for the bottleneck capacity for test cases with time-varying bottleneck capacities. All bottleneck capacities will be specified as a ratio with respect to the reference capacity value.

+ One-way propagation delay: describes the end-to-end latency along the path when network queues are empty, i.e., the time it takes for a packet to go from the sender to the receiver without encountering any queuing delay.

+ Maximum end-to-end jitter: defines the maximum jitter that can be observed along the path.

+ Bottleneck queue type: for example, Droptail, FQ-CoDel, or PIE.

+ Bottleneck queue size: defines the size of queue in terms of queuing time when the queue is full (in milliseconds).

+ Path loss ratio: characterizes the non-congested, additive, losses to be generated on the end-to-end path. MUST describe the loss pattern or loss model used to generate the losses.

* Application-related: defines the traffic source behavior for implementing the test case

+ Media traffic Source: defines the characteristics of the media sources. When using more than one media source, the different attributes are enumerated separately for each different media source.

- Media type: Video/Voice

- Media flow direction: forward, backward or both.

- Number of media sources: defines the total number of media sources

- Media codec: Constant Bit Rate (CBR) or Variable Bit Rate (VBR)

- Media source behavior: describes the media encoder behavior. It defines the main parameters that affect the adaptation behavior. This may include but is not limited to:

  o Adaptability: describes the adaptation options. For example, in the case of video it defines the following ranges of adaptation: bit rate, frame rate, video resolution. Similarly, in the case of voice, it defines the range of bit rate adaptation, the sampling rate variation, and the variation in packetization interval.

  o Output variation : for a VBR encoder it defines the encoder output variation from the average target rate over a particular measurement interval. For example, on average the encoder output may vary between 5% to 15% above or below the average target bit rate when measured over a 100 ms time window. The time interval over which the variation is specified must be provided.

  o Responsiveness to a new bit rate request: the lag in time between a new bit rate request from the congestion control algorithm and actual rate changes in encoder output. Depending on the encoder, this value may be specified in absolute time (e.g. 10ms to 1000ms) or other appropriate metric (e.g. next frame interval time).

  More detailed discussions on expected media source behavior, including those from synthetic video traffic sources, is at [I-D.ietf-rmcat-video-traffic-model].

- Media content: describes the chosen media sequences; For example, test sequences are available at: [xiph-seq] and [HEVC-seq].

- Media timeline: describes the point when the media source is introduced and removed from the testbed. For example, the media source may start transmitting immediately when the test case begins, or after a few seconds.

- Startup behavior: the media starts at a defined bit rate, which may be the minimum, maximum bit rate, or a value in between (in Kbps).

+ Competing traffic source: describes the characteristics of the competing traffic source, the different types of competing flows are enumerated in [I-D.ietf-rmcat-eval-criteria].

  - Traffic direction: forward, backward or both.

  - Type of sources: defines the types of competing traffic sources. Types of competing traffic flows are listed in [I-D.ietf-rmcat-eval-criteria]. For example, the number of TCP flows connected to a web browser, the mean size and distribution of the content downloaded.

  - Number of sources: defines the total number of competing sources of each media type per traffic direction.

  - Congestion control: enumerates the congestion control used by each type of competing traffic.

  - Traffic timeline: describes when the competing traffic starts and ends in the test case.

* Additional attributes: describes attributes essential for implementing a test case which are not included in the above structure. These attributes MUST be well defined, so that the other implementers of that particular test case are able to implement it easily.

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

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

4. Recommended Evaluation Settings

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

4.1. Evaluation metrics

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

1. Flow level:

    A. End-to-end delay for the congestion controlled media flow.

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

    C. Packet losses observed at the receiving endpoint.

    D. Feedback message overhead.

    E. Convergence time - time to reach steady state for the congestion controlled media flow(s).

2. Transport level:

    A. Bandwidth utilization.

    B. Queue length (milliseconds at specified path capacity):

        + average over the length of the session.

        + 5 and 95 percentile.

        + median, maximum, minimum.

4.2. Path characteristics

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

- Path direction: forward and backward.

- Reference bottleneck capacity: 1Mbps.

- One-Way propagation delay: 50ms. Implementers are encouraged to run the experiment with additional propagation delays mentioned in [I-D.ietf-rmcat-eval-criteria]

- Maximum end-to-end jitter: 30ms. Jitter models are described in [I-D.ietf-rmcat-eval-criteria]

- Bottleneck queue type: Drop tail. Implementers are encouraged to run the experiment with other AQM schemes, such as FQ-CoDel and PIE.

- Bottleneck queue size: 300ms.

- Path loss ratio: 0%.

Examples of additional network parameters are discussed in [I-D.ietf-rmcat-eval-criteria].

For test cases involving time-varying bottleneck capacity, all capacity values are specified as a ratio with respect to a reference capacity value, so as to allow flexible scaling of capacity values along with media source rate range. There exist two different mechanisms for inducing path capacity variation: a) by explicitly modifying the value of physical link capacity; or b) by introducing background non-adaptive UDP traffic with time-varying traffic rate. Implementers are encouraged to run the experiments with both mechanisms for test cases specified in Section 5.1, Section 5.2, and Section 5.3.

4.3. Media source

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

- Media type: Video
  - Media codec: VBR
  - Media source behavior:
    - Adaptability:
      - Bit rate range: 150 Kbps - 1.5 Mbps. In real-life applications the bit rate range can vary a lot depending on the provided service, for example, the maximum bit rate can be up to 4Mbps. However, for running tests to evaluate the congestion control algorithms it is more important to have a look at how they are reacting to certain amount of bandwidth change. Also it is possible that the media traffic generator used in a particular simulator or testbed is not capable of generating higher bit rate. Hence we have selected a suitable bit rate range typical of consumer-grade video conferencing applications in designing the test case. If a different bit rate range is used in the test cases, then the end-

to-end path capacity values will also need to be scaled accordingly.

- Frame resolution: 144p - 720p (or 1080p). This resolution range is selected based on the bit rate range. If a different bit rate range is used in the test cases then the frame resolution range also need to be selected suitably.

- Frame rate: 10fps - 30fps. This frame rate range is selected based on the bit rate range. If a different bit rate range is used in the test cases then the frame rate range also need to be adjusted suitably.

+ Variation from target bit rate: +/-5%. Unless otherwise specified in the test case(s), bit rate variation SHOULD be calculated over one (1) second period of time.

+ Responsiveness to new bit rate request: 100ms

* Media content: The media content should represent a typical video conversational scenario with head and shoulder movement. We recommend to use Foreman video sequence.

* Media startup behavior: 150Kbps. It should be noted that applications can use smart ways to select an optimal startup bit rate value for a certain network condition. In such cases the candidate proposals MAY show the effectiveness of such smart approach as an additional information for the evaluation process.

o Media type: Audio

* Media codec: CBR

* Media bit rate: 20Kbps

5. Basic Test Cases

5.1. Variable Available Capacity with a Single Flow

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

- an intermediate bottleneck

- change in available capacity (e.g., due to interface change, routing change, abrupt arrival/departure of background non-adaptive traffic).

- maximum media bit rate is greater than link capacity. In this case, the application will attempt to ramp up to its maximum bit rate, since the link capacity is limited to a value lower, the congestion control scheme is expected to stabilize the sending bit rate close to the available bottleneck capacity.

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

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

Evaluation metrics : as described in Section 4.1.

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

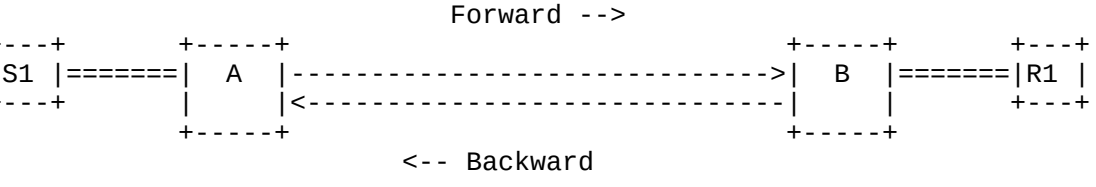


Figure 2: Testbed Topology for Limited Link Capacity

Testbed attributes:

- Test duration: 100s

- Path characteristics: as described in Section 4.2

- Application-related:
  - Media Traffic:
    - Media type: Video
      - Media direction: forward.
      - Number of media sources: one (1)
      - Media timeline:
        - Start time: 0s.
        - End time: 99s.
    - Media type: Audio
      - Media direction: forward.
      - Number of media sources: one (1)
      - Media timeline:
        - Start time: 0s.
        - End time: 99s.
  - Competing traffic:
    - Number of sources : zero (0)
- Test Specific Information:
  - One-way propagation delay: [50 ms, 100 ms]. on the forward path direction
  - This test uses bottleneck path capacity variation as listed in Table 1
  - When using background non-adaptive UDP traffic to induce time-varying bottleneck , the physical path capacity remains at 4Mbps and the UDP traffic source rate changes over time as (4-x)Mbps, where x is the bottleneck capacity specified in Table 1

Variation pattern index	Path direction	Start time	Path capacity ratio
One	Forward	0s	1.0
Two	Forward	40s	2.5
Three	Forward	60s	0.6
Four	Forward	80s	1.0

Table 1: Path capacity variation pattern for forward direction

5.2. Variable Available Capacity with Multiple Flows

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

Expected behavior: the candidate algorithm is expected to detect the variation in available capacity and adapt the media stream(s) accordingly. The flows stabilize around their maximum bit rate as the maximum link capacity is large enough to accommodate the flows. When the available capacity drops, the flows adapt by decreasing their sending bit rate, and when congestion disappears, the flows are again expected to ramp up.

Evaluation metrics : as described in Section 4.1.

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

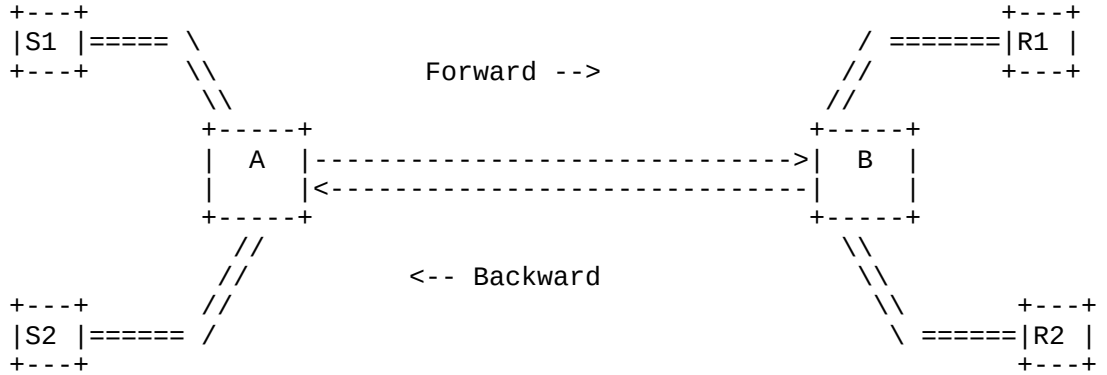


Figure 3: Testbed Topology for Variable Available Capacity

Testbed attributes:

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

Test Specific Information: This test uses path capacity variation as listed in Table 2 with a corresponding end time of 125 seconds. The reference bottleneck capacity is 2Mbps. When using background non-adaptive UDP traffic to induce time-varying bottleneck for congestion controlled media flows, the physical path capacity is 4Mbps and the UDP traffic source rate changes over time as (4-x)Mbps, where x is the bottleneck capacity specified in Table 2.

Variation pattern index	Path direction	Start time	Path capacity ratio
One	Forward	0s	2.0
Two	Forward	25s	1.0
Three	Forward	50s	1.75
Four	Forward	75s	0.5
Five	Forward	100s	1.0

Table 2: Path capacity variation pattern for forward direction

5.3. Congested Feedback Link with Bi-directional Media Flows

Real-time interactive media uses RTP hence it is assumed that RTCP, RTP header extension or such would be used by the congestion control algorithm in the backchannel. Due to asymmetric nature of the link between communicating peers it is possible for a participating peer to not receive such feedback information due to an impaired or congested backchannel (even when the forward channel might not be impaired). This test case is designed to observe the candidate congestion control behavior in such an event.

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

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

Evaluation metrics : as described in Section 4.1.

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

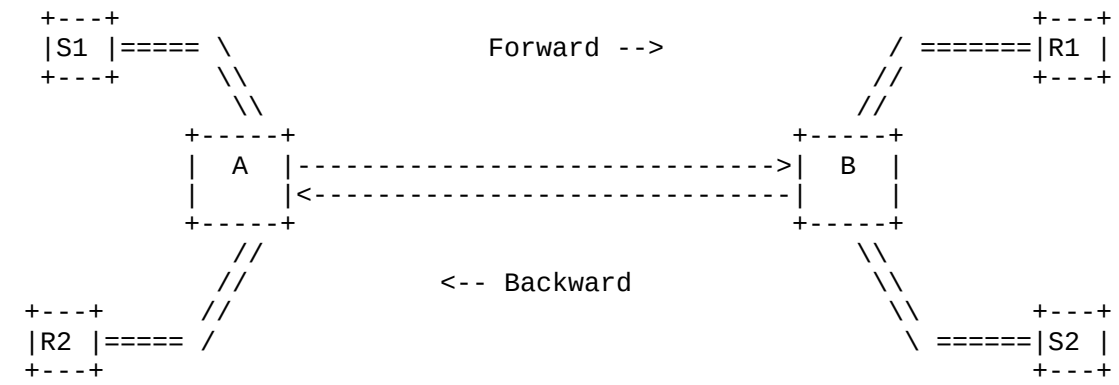


Figure 4: Testbed Topology for Congested Feedback Link

Testbed attributes:

- Test duration: 100s
- Path characteristics:
  - Reference bottleneck capacity: 1Mbps.
- Application-related:
  - Media Source:
    - Media type: Video
      - Media direction: forward and backward
      - Number of media sources: two (2)
      - Media timeline:
        - Start time: 0s.
        - End time: 99s.

+ Media type: Audio

- Media direction: forward and backward

- Number of media sources: two (2)

- Media timeline:

o Start time: 0s.

o End time: 99s.

* Competing traffic:

+ Number of sources : zero (0)

o Test Specific Information: this test uses path capacity variations to create congested feedback link. Table 3 lists the variation patterns applied to the forward path and Table 4 lists the variation patterns applied to the backward path. When using background non-adaptive UDP traffic to induce time-varying bottleneck for congestion controlled media flows, the physical path capacity is 4Mbps for both directions and the UDP traffic source rate changes over time as (4-x)Mbps in each direction, where x is the bottleneck capacity specified in Table 4.

Variation pattern index	Path direction	Start time	Path capacity ratio
One	Forward	0s	2.0
Two	Forward	20s	1.0
Three	Forward	40s	0.5
Four	Forward	60s	2.0

Table 3: Path capacity variation pattern for forward direction

Variation pattern index	Path direction	Start time	Path capacity ratio
One	Backward	0s	2.0
Two	Backward	35s	0.8
Three	Backward	70s	2.0

Table 4: Path capacity variation pattern for backward direction

5.4. Competing Media Flows with same Congestion Control Algorithm

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

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

Evaluation metrics : as described in Section 4.1.

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

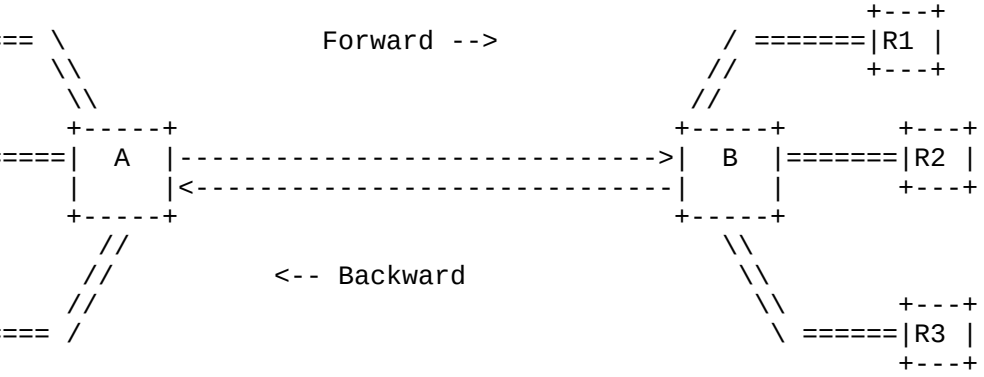


Figure 5: Testbed Topology for Multiple congestion controlled media Flows

Testbed attributes:

- Test duration: 120s
- Path characteristics:
  - Reference bottleneck capacity: 3.5Mbps
  - Path capacity ratio: 1.0
- Application-related:
  - Media Source:
    - Media type: Video
      - Media direction: forward.
      - Number of media sources: three (3)
      - Media timeline: new media flows are added sequentially, at short time intervals. See test specific setup below.
    - Media type: Audio
      - Media direction: forward.
      - Number of media sources: three (3)
      - Media timeline: new media flows are added sequentially, at short time intervals. See test specific setup below.
  - Competing traffic:
    - Number of sources : zero (0)
- Test Specific Information: Table 5 defines the media timeline for both media type.

Flow ID	Media type	Start time	End time
1	Video	0s	119s
2	Video	20s	119s
3	Video	40s	119s
4	Audio	0s	119s
5	Audio	20s	119s
6	Audio	40s	119s

Table 5: Media Timeline for Video and Audio media sources

5.5. Round Trip Time Fairness

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

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

Evaluation metrics : as described in Section 4.1.

Testbed Topology: Five (5) media sources S1,S2,..,S5 are connected to their corresponding media sinks R1,R2,..,R5. The media traffic is transported over the forward path and corresponding feedback/control traffic is transported over the backward path. The topology is the same as in Section 5.4. The end-to-end path delays are: 10ms for S1-R1, 25ms for S2-R2, 50ms for S3-R3, 100ms for S4-R4, and 150ms S5-R5, respectively.

Testbed attributes:

- Test duration: 300s
- Path characteristics:
  - One-Way propagation delay for each flow: 10ms, 25ms, 50ms, 100ms, 150ms.
- Application-related:

* Media Source:

  + Media type: Video

    - Media direction: forward

    - Number of media sources: five (5)

    - Media timeline: new media flows are added sequentially, at short time intervals. See test specific setup below.

  + Media type: Audio

    - Media direction: forward.

    - Number of media sources: five (5)

    - Media timeline: new media flows are added sequentially, at short time intervals. See test specific setup below.

* Competing traffic:

  + Number of sources : zero (0)

o Test Specific Information: Table 6 defines the media timeline for both media type.

Flow IF	Media type	Start time	End time
1	Video	0s	299s
2	Video	10s	299s
3	Video	20s	299s
4	Video	30s	299s
5	Video	40s	299s
6	Audio	0	299s
7	Audio	10s	299s
8	Audio	20s	299s
9	Audio	30s	299s
10	Audio	40s	299s

Table 6: Media Timeline for Video and Audio media sources

5.6. Media Flow Competing with a Long TCP Flow

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

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

Evaluation metrics : following metrics in addition to as described in Section 4.1.

1. Flow level:

    A. TCP throughput.

    B. Loss for the TCP flow

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

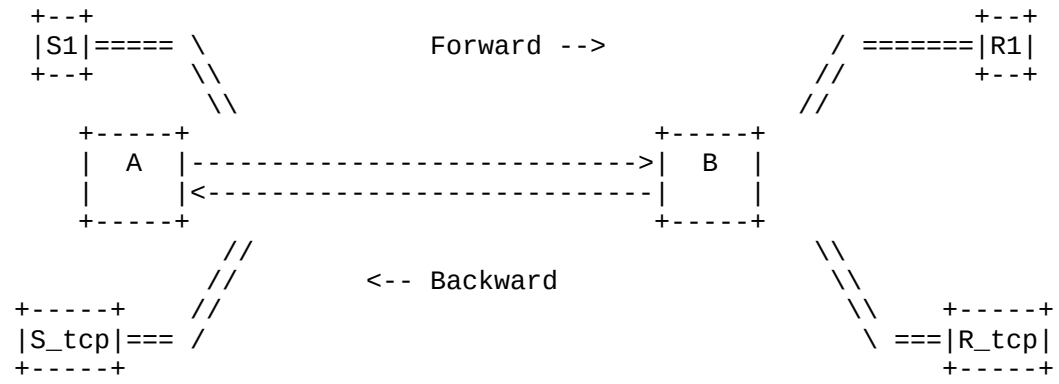


Figure 6: Testbed Topology for TCP vs congestion controlled media Flows

Testbed attributes:

- Test duration: 120s
- Path characteristics:
  - Reference bottleneck capacity: 2Mbps
  - Path capacity ratio: 1.0
  - Bottleneck queue size: [300ms, 1000ms]
- Application-related:
  - Media Source:
    - Media type: Video
      - Media direction: forward
      - Number of media sources: one (1)
      - Media timeline:
        - Start time: 5s.
        - End time: 119s.
    - Media type: Audio
      - Media direction: forward

- Number of media sources: one (1)
- Media timeline:
  - Start time: 5s.
  - End time: 119s.

* Additionally, implementers are encouraged to run the experiment with multiple media sources.
* Competing traffic:
  + Number and Types of sources : one (1) and long-lived TCP
  + Traffic direction : forward
  + Congestion control: default TCP congestion control[RFC5681].
  + Traffic timeline:
    - Start time: 0s.
    - End time: 119s.

o Test Specific Information: none

5.7. Media Flow Competing with Short TCP Flows

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

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

Evaluation metrics : following metrics in addition to as described in Section 4.1.

1. Flow level:

    A. Variation in the sending rate of the TCP flow.

    B. TCP throughput.

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

Testbed attributes:

- Test duration: 300s
- Path characteristics:
  - Reference bottleneck capacity: 2.0Mbps
  - Path capacity ratio: 1.0
- Application-related:
  - Media source:
    - Media type: Video
      - Media direction: forward
      - Number of media sources: two (2)
      - Media timeline:
        - Start time: 5s.
        - End time: 299s.
    - Media type: Audio
      - Media direction: forward
      - Number of media sources: two (2)
      - Media timeline:
        - Start time: 5s.
        - End time: 299s.
  - Competing traffic:

+ Number and Types of sources : ten (10), short-lived TCP flows.

+ Traffic direction : forward

+ Congestion algorithm: default TCP Congestion control [RFC5681].

+ Traffic timeline: each short TCP flow is modeled as a sequence of file downloads interleaved with idle periods. See test specific setup. Not all short TCP flows start at the same time, 2 of them start in the ON state while rest on the 8 flows start in an OFF stats. The model for the idle times for the OFF state is discussed in [I-D.ietf-rmcat-eval-criteria].

o Test Specific Information:

* Short-TCP traffic model:

+ File sizes: uniform distribution between 100KB to 1MB

+ Idle period: the duration of the OFF state is derived from an exponential distribution with the mean value of 10 seconds.

5.8. Media Pause and Resume

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

Evaluation metrics : following metrics in addition to as described in Section 4.1.

1. Flow level:

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

Testbed Topology: Same as test case defined in Section 5.4

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

- Other test specific setup:
  - Media flow timeline:
    - Flow ID: one (1)
    - Start time: 0s
    - Flow duration: 119s
    - Pause time: not required
    - Resume time: not required
  - Media flow timeline:
    - Flow ID: two (2)
    - Start time: 0s
    - Flow duration: 119s
    - Pause time: at 40s
    - Resume time: at 60s
  - Media flow timeline:
    - Flow ID: three (3)
    - Start time: 0s
    - Flow duration:119s

+ Pause time: not required

+ Resume time: not required

6. Other potential test cases

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

6.1. Media Flows with Priority

In this test case media flows will have different priority levels. This will be an extension of Section 5.4 where the same test will be run with different priority levels imposed on each of the media flows. For example, the first flow (S1) is assigned a priority of 2 whereas the remaining two flows (S2 and S3) are assigned a priority of 1. The candidate algorithm MUST reflect the relative priorities assigned to each media flow. In the previous example, the first flow (S1) MUST arrive at a steady-state rate approximately twice of that of the other two flows (S2 and S3).

The candidate algorithm can use a coupled congestion control mechanism for the bandwidth distribution according to the respective media flow priority.

6.2. Explicit Congestion Notification Usage

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

6.3. Multiple Bottlenecks

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

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

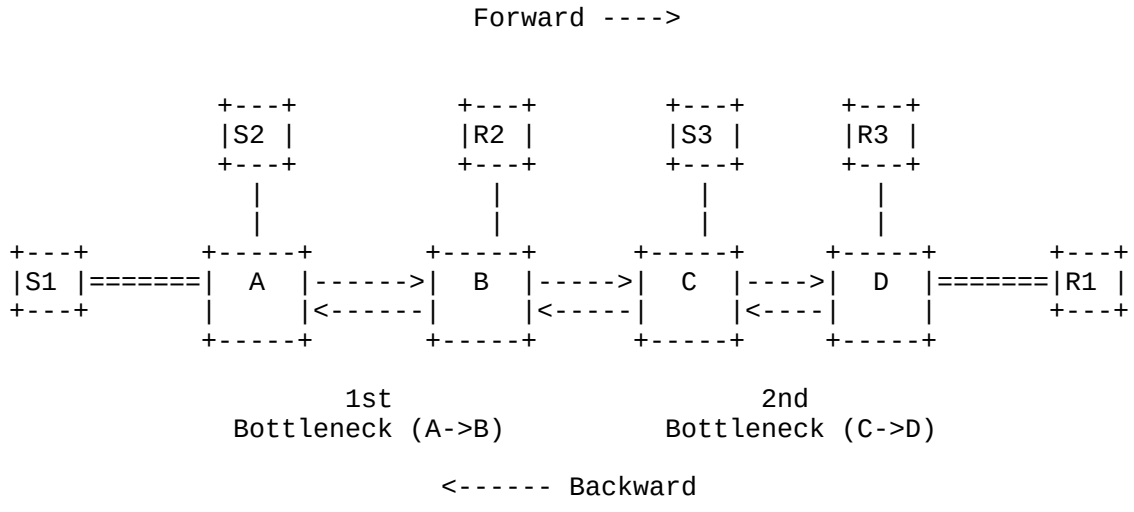


Figure 7: Testbed Topology for Multiple Bottlenecks

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

Testbed attributes:

- Test duration: 300s
- Path characteristics:
  - Reference bottleneck capacity: 2Mbps.
  - Path capacity ratio between A and B: 1.0
  - Path capacity ratio between B and C: 4.0.
  - Path capacity ratio between C and D: 0.75.

* One-Way propagation delay:
  1. Between S1 and R1: 100ms
  2. Between S2 and R2: 40ms
  3. Between S3 and R3: 40ms

o Application-related:
* Media Source:
  + Media type: Video
    - Media direction: Forward
    - Number of media sources: Three (3)
    - Media timeline:
      o Start time: 0s.
      o End time: 299s.
  + Media type: Audio
    - Media direction: Forward
    - Number of media sources: Three (3)
    - Media timeline:
      o Start time: 0s.
      o End time: 299s.
* Competing traffic:
  + Number of sources : Zero (0)

7. Wireless Access Links

Additional wireless network (both cellular network and WiFi network) specific test cases are defined in [I-D.ietf-rmcat-wireless-tests].

8. Security Considerations

Security issues have not been discussed in this memo.

9. IANA Considerations

There are no IANA impacts in this memo.

10. Acknowledgements

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

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

11. References

11.1. Normative References

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

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

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <http://www.rfc-editor.org/info/rfc3551>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <http://www.rfc-editor.org/info/rfc5506>.

[I-D.ietf-rmcat-eval-criteria]
Singh, V. and J. Ott, "Evaluating Congestion Control for Interactive Real-time Media", draft-ietf-rmcat-eval-criteria-04 (work in progress), October 2015.

[I-D.ietf-rmcat-wireless-tests]
Sarker, Z., Johansson, I., Zhu, X., Fu, J., Tan, W., and M. Ramalho, "Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks", draft-ietf-rmcat-wireless-tests-01 (work in progress), November 2015.

[I-D.ietf-rmcat-video-traffic-model]
Zhu, X., Cruz, S., and Z. Sarker, "Modeling Video Traffic Sources for RMCAT Evaluations", draft-ietf-rmcat-video-traffic-model-00 (work in progress), January 2016.

11.2. Informative References

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

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

[xiph-seq]
Xiph.org, , "Video Test Media", http://media.xiph.org/video/derf/ .

[HEVC-seq]
HEVC, , "Test Sequences", http://www.netlab.tkk.fi/~varun/test_sequences/ .

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Luleae, SE 977 53
Sweden

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

Varun Singh
Nemu Dialogue Systems Oy
Runeberginkatu 4c A 4
Helsinki 00100
Finland

Email: varun.singh@iki.fi
URI: http://www.callstats.io/

Xiaoqing Zhu
Cisco Systems
12515 Research Blvd
Austing, TX 78759
USA

Email: xiaoqzhu@cisco.com

Michael A. Ramalho
Cisco Systems, Inc.
6310 Watercrest Way Unit 203
Lakewood Ranch, FL 34202-5211
USA

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

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

July 8, 2016

A Google Congestion Control Algorithm for Real-Time Communication
draft-ietf-rmcat-gcc-02

Abstract

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

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

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Table of Contents

1. Introduction

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

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

- The media is usually encoded in forms that cannot be quickly changed to accommodate varying bandwidth, and bandwidth requirements can often be changed only in discrete, rather large steps

- The participants may have certain specific wishes on how to respond - which may not be reducing the bandwidth required by the flow on which congestion is discovered

- The encodings are usually sensitive to packet loss, while the real-time requirement precludes the repair of packet loss by retransmission

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

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

1.1. Mathematical notation conventions

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

The following notational conventions are used:

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

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

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

2. System model

The following elements are in the system:

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

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

3. Feedback and extensions

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

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

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

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

4. Sending Engine

Pacing is used to actuate the target bitrate computed by the controllers.

When media encoder produces data, this is fed into a Pacer queue. The Pacer sends a group of packets to the network every burst_time interval. RECOMMENDED value for burst_time is 5 ms. The size of a group of packets is computed as the product between the target bitrate and the burst_time.

5. Delay-based control

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

5.1. Arrival-time model

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

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

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

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

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

We can model the inter-group delay variation as:

$$d(i) = w(i)$$

Here, w(i) is a sample from a stochastic process W, which is a function of the link capacity, the current cross traffic, and the current sent bitrate. We model W as a white Gaussian process. If we are over-using the channel we expect the mean of w(i) to increase, and if a queue on the network path is being emptied, the mean of w(i) will decrease; otherwise the mean of w(i) will be zero.

Breaking out the mean, m(i), from w(i) to make the process zero mean, we get

Equation 1

$$d(i) = m(i) + v(i)$$

The noise term v(i) represents network jitter and other delay effects not captured by the model.

5.2. Pre-filtering

The pre-filtering aims at handling delay transients caused by channel outages. During an outage, packets being queued in network buffers, for reasons unrelated to congestion, are delivered in a burst when the outage ends.

The pre-filtering merges together groups of packets that arrive in a burst. Packets are merged in the same group if one of these two conditions holds:

- A sequence of packets which are sent within a burst_time interval constitute a group.
- A Packet which has an inter-arrival time less than burst_time and an inter-group delay variation d(i) less than 0 is considered being part of the current group of packets.

5.3. Arrival-time filter

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

Let m(i) be the estimate at time i

We model the state evolution from time i to time i+1 as

$$m(i+1) = m(i) + u(i)$$

where u(i) is the state noise that we model as a stationary process with Gaussian statistic with zero mean and variance

$$q(i) = E\{u(i)^2\}$$

q(i) is RECOMMENDED equal to 10^-3

Given equation 1 we get

$$d(i) = m(i) + v(i)$$

where v(i) is zero mean white Gaussian measurement noise with variance var_v = E{v(i)^2}

The Kalman filter recursively updates our estimate m_hat(i) as

$$z(i) = d(i) - m\_hat(i-1)$$

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

$$k(i) = \frac{e(i-1) + q(i)}{var\_v\_hat(i) + (e(i-1) + q(i))}$$

$$e(i) = (1 - k(i)) * (e(i-1) + q(i))$$

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

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

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

where f_max = max {1/(T(j) - T(j-1))} for j in i-K+1,...,i is the highest rate at which the last K packet groups have been received and chi is a filter coefficient typically chosen as a number in the interval [0.1, 0.001]. Since our assumption that v(i) should be zero mean WGN is less accurate in some cases, we have introduced an additional outlier filter around the updates of var_v_hat. If z(i) > 3*sqrt(var_v_hat) the filter is updated with 3*sqrt(var_v_hat) rather than z(i). For instance v(i) will not be white in situations where packets are sent at a higher rate than the channel capacity, in which case they will be queued behind each other.

5.4. Over-use detector

The inter-group delay variation estimate m(i), obtained as the output of the arrival-time filter, is compared with a threshold del_var_th(i). An estimate above the threshold is considered as an indication of over-use. Such an indication is not enough for the detector to signal over-use to the rate control subsystem. A definitive over-use will be signaled only if over-use has been detected for at least overuse_time_th milliseconds. However, if m(i) < m(i-1), over-use will not be signaled even if all the above conditions are met. Similarly, the opposite state, under-use, is detected when m(i) < -del_var_th(i). If neither over-use nor under-use is detected, the detector will be in the normal state.

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

The reason is that, by using a larger value of del_var_th, a larger queuing delay can be tolerated, whereas with a small del_var_th, the over-use detector quickly reacts to a small increase in the offset estimate m(i) by generating an over-use signal that reduces the delay-based estimate of the available bandwidth A_hat (see Section 4.4). Thus, it is necessary to dynamically tune the threshold del_var_th to get good performance in the most common scenarios, such as when competing with loss-based flows.

For this reason, we propose to vary the threshold del_var_th(i) according to the following dynamic equation:

```
del_var_th(i) =
        del_var_th(i-1) + (t(i)-t(i-1)) * K(i) * (|m(i)|-del_var_th(i-1))
```

with K(i)=K_d if |m(i)| < del_var_th(i-1) or K(i)=K_u otherwise. The rationale is to increase del_var_th(i) when m(i) is outside of the range [-del_var_th(i-1),del_var_th(i-1)], whereas, when the offset estimate m(i) falls back into the range, del_var_th is decreased. In this way when m(i) increases, for instance due to a TCP flow entering the same bottleneck, del_var_th(i) increases and avoids the uncontrolled generation of over-use signals which may lead to starvation of the flow controlled by the proposed algorithm [Pv13]. Moreover, del_var_th(i) SHOULD NOT be updated if this condition holds:

```
|m(i)| - del_var_th(i) > 15
```

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

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

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

5.5. Rate control

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

As soon as over-use has been detected, the available bandwidth estimated by the delay-based controller is decreased. In this way we get a recursive and adaptive estimate of the available bandwidth.

In this document we make the assumption that the rate control subsystem is executed periodically and that this period is constant.

The rate control subsystem has 3 states: Increase, Decrease and Hold. "Increase" is the state when no congestion is detected; "Decrease" is the state where congestion is detected, and "Hold" is a state that waits until built-up queues have drained before going to "increase" state.

The state transitions (with blank fields meaning "remain in state") are:

\ State / Signal	Hold	Increase	Decrease
Over-use	Decrease	Decrease	
Normal	Increase		Hold
Under-use		Hold	Hold

The subsystem starts in the increase state, where it will stay until over-use or under-use has been detected by the detector subsystem. On every update the delay-based estimate of the available bandwidth is increased, either multiplicatively or additively, depending on its current state.

The system does a multiplicative increase if the current bandwidth estimate appears to be far from convergence, while it does an additive increase if it appears to be closer to convergence. We assume that we are close to convergence if the currently incoming bitrate, R_hat(i), is close to an average of the incoming bitrates at

the time when we previously have been in the Decrease state. "Close" is defined as three standard deviations around this average. It is RECOMMENDED to measure this average and standard deviation with an exponential moving average with the smoothing factor 0.95, as it is expected that this average covers multiple occasions at which we are in the Decrease state. Whenever valid estimates of these statistics are not available, we assume that we have not yet come close to convergence and therefore remain in the multiplicative increase state.

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

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

```
R_hat(i) = 1/T * sum(L(j)) for j from 1 to N(i)
```

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

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

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

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

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

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

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

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

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

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

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

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

When the detector signals under-use to the rate control subsystem, we know that queues in the network path are being emptied, indicating that our available bandwidth estimate A_hat is lower than the actual available bandwidth. Upon that signal the rate control subsystem will enter the hold state, where the receive-side available bandwidth estimate will be held constant while waiting for the queues to stabilize at a lower level - a way of keeping the delay as low as possible. This decrease of delay is wanted, and expected, immediately after the estimate has been reduced due to over-use, but can also happen if the cross traffic over some links is reduced.

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

5.6. Parameters settings

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

Table 1: RECOMMENDED values for delay based controller

Table 1

6. Loss-based control

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

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

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

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

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

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

The loss-based estimate As_hat is compared with the delay-based estimate A_hat. The actual sending rate is set as the minimum between As_hat and A_hat.

We motivate the packet loss thresholds by noting that if the transmission channel has a small amount of packet loss due to over-use, that amount will soon increase if the sender does not adjust his bitrate. Therefore we will soon enough reach above the 10% threshold and adjust As_hat(i). However, if the packet loss ratio does not increase, the losses are probably not related to self-inflicted congestion and therefore we should not react on them.

7. Interoperability Considerations

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

8. Implementation Experience

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

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

9. Further Work

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

Work that can be done on this basis includes:

- Considerations of integrated loss control: How loss and delay control can be better integrated, and the loss control improved.

- Considerations of locus of control: evaluate the performance of having all congestion control logic at the sender, compared to splitting logic between sender and receiver.

- Considerations of utilizing ECN as a signal for congestion estimation and link over-use detection.

10. IANA Considerations

This document makes no request of IANA.

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

11. Security Considerations

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

In this case, the control information is carried inside RTP, and can be protected against modification or message insertion using SRTP, just as for the media. Given that timestamps are carried in the RTP header, which is not encrypted, this is not protected against disclosure, but it seems hard to mount an attack based on timing information only.

12. Acknowledgements

Thanks to Randell Jesup, Magnus Westerlund, Varun Singh, Tim Panton, Soo-Hyun Choo, Jim Gettys, Ingemar Johansson, Michael Welzl and others for providing valuable feedback on earlier versions of this draft.

13. References

13.1. Normative References

[I-D.alvestrand-rmcat-remb]
Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", draft-alvestrand-rmcat-remb-03 (work in progress), October 2013.

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

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

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

13.2. Informative References

[Pv13] De Cicco, L., Carlucci, G., and S. Mascolo, "Understanding the Dynamic Behaviour of the Google Congestion Control", Packet Video Workshop , December 2013.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

Appendix A. Change log

A.1. Version -00 to -01

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

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

A.2. Version -01 to -02

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

A.3. Version -02 to -03

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

A.4. rtcweb-03 to rmcat-00

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

A.5. rmcat -00 to -01

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

A.6. rmcat -01 to -02

- Added Luca De Cicco and Saverio Mascolo as authors.
- Extended the "Over-use detector" section with new technical details on how to dynamically tune the offset del_var_th for improved fairness properties.

- Added reference to a paper analyzing the behavior of the proposed algorithm.

A.7. rmcat -02 to -03

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

A.8. ietf-rmcat -00 to ietf-rmcat -01

- Arrival-time filter converted from a two dimensional Kalman filter to a scalar Kalman filter.
- The use of the TFRC equation was removed from the loss-based controller, as it turned out to have little to no effect in practice.

A.9. ietf-rmcat -01 to ietf-rmcat -02

- Added a section which better describes the pre-filtering algorithm.

Authors' Addresses

Stefan Holmer
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: holmer@google.com

Henrik Lundin
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: hlundin@google.com

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

Email: gaetano.carlucci@poliba.it

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

Email: l.decicco@poliba.it

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

Email: mascolo@poliba.it

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: March 8, 2020

X. Zhu
R. Pan
M. Ramalho
S. Mena
Cisco Systems
September 5, 2019

NADA: A Unified Congestion Control Scheme for Real-Time Media
draft-ietf-rmcat-nada-13

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 8, 2020.

Table of Contents

1. Introduction

Interactive real-time media applications introduce a unique set of challenges for congestion control. Unlike TCP, the mechanism used for real-time media needs to adapt quickly to instantaneous bandwidth changes, accommodate fluctuations in the output of video encoder rate control, and cause low queuing delay over the network. An ideal scheme should also make effective use of all types of congestion signals, including packet loss, queuing delay, and explicit congestion notification (ECN) [RFC3168] markings. The requirements for the congestion control algorithm are outlined in [I-D.ietf-rmcat-cc-requirements]. It highlights that the desired congestion control scheme should avoid flow starvation and attain a reasonable fair share of bandwidth when competing against other flows, adapt quickly, and operate in a stable manner.

This document describes an experimental congestion control scheme called network-assisted dynamic adaptation (NADA). The design of NADA benefits from explicit congestion control signals (e.g., ECN markings) from the network, yet also operates when only implicit congestion indicators (delay and/or loss) are available. Such a unified sender behavior distinguishes NADA from other congestion control schemes for real-time media. In addition, its core congestion control algorithm is designed to guarantee stability for path round-trip-times (RTTs) below a prescribed bound (e.g., 250ms with default parameter choices). It further supports weighted bandwidth sharing among competing video flows with different priorities. The signaling mechanism consists of standard RTP timestamp [RFC3550] and RTCP feedback reports. The definition of the desired RTCP feedback message is described in detail in [I-D.ietf-avtcore-cc-feedback-message] so as to support the successful operation of several congestion control schemes for real-time interactive media.

2. Terminology

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

3. System Overview

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

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

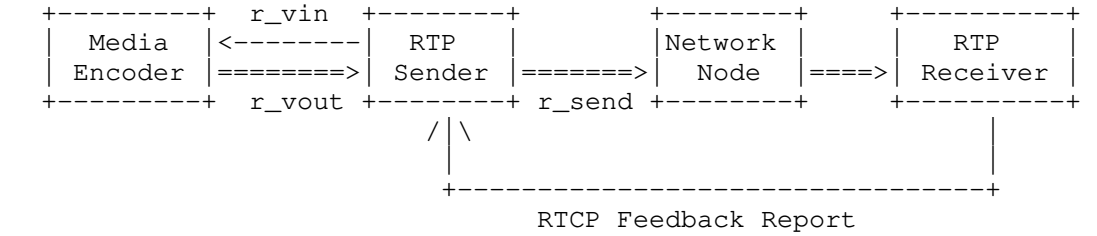


Figure 1: System Overview

- Media encoder with rate control capabilities. It encodes raw media (audio and video) frames into a compressed bitstream which is later packetized into RTP packets. As discussed in [RFC8593], the actual output rate from the encoder r_vout may fluctuate around the target r_vin. Furthermore, it is possible that the encoder can only react to bit rate changes at rather coarse time intervals, e.g., once every 0.5 seconds.

- RTP sender: responsible for calculating the NADA reference rate based on network congestion indicators (delay, loss, or ECN marking reports from the receiver), for updating the video encoder with a new target rate r_vin, and for regulating the actual sending rate r_send accordingly. The RTP sender also generates a sending timestamp for each outgoing packet.

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

- Network node with several modes of operation. The system can work with the default behavior of a simple drop tail queue. It can also benefit from advanced AQM features such as PIE [RFC8033], FQ-CoDel [RFC8290], ECN marking based on RED [RFC7567], and PCN marking using a token bucket algorithm ([RFC6660]). Note that network node operation is out of control for the design of NADA.

4. Core Congestion Control Algorithm

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

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

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

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

4.1. Mathematical Notations

This section summarizes the list of variables and parameters used in the NADA algorithm. Figure 3 also includes the default values for choosing the algorithm parameters either to represent a typical setting in practical applications or based on theoretical and simulation studies. See Section 6.3 for some of the discussions on the impact of parameter values. Additional studies in real-world settings suggested in Section 8 could gather further insight on how to choose and adapt these parameter values in practical deployment.

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

Figure 2: List of variables.

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

TAU	Upper bound of RTT in gradual rate update calculation	500ms
DELTA	Target feedback interval	100ms
LOGWIN	Observation window in time for calculating packet summary statistics at receiver	500ms
QEPS	Threshold for determining queuing delay build up at receiver	10ms
DFILT	Bound on filtering delay	120ms
GAMMA_MAX	Upper bound on rate increase ratio for accelerated ramp-up	0.5
QBOUND	Upper bound on self-inflicted queuing delay during ramp up	50ms
MULTILOSS	Multiplier for self-scaling the expiration threshold of the last observed loss (loss_exp) based on measured average loss interval (loss_int)	7.0
QTH	Delay threshold for invoking non-linear warping	50ms
LAMBDA	Scaling parameter in the exponent of non-linear warping	0.5
PLRREF	Reference packet loss ratio	0.01
PMRREF	Reference packet marking ratio	0.01
DLOSS	Reference delay penalty for loss when packet loss ratio is at PLRREF	10ms
DMARK	Reference delay penalty for ECN marking when packet marking is at PMRREF	2ms
FPS	Frame rate of incoming video	30
BETA_S	Scaling parameter for modulating outgoing sending rate	0.1
BETA_V	Scaling parameter for modulating video encoder target rate	0.1
ALPHA	Smoothing factor in exponential smoothing of packet loss and marking ratios	0.1

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

4.2. Receiver-Side Algorithm

The receiver-side algorithm can be outlined as below:

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

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

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

In order for a delay-based flow to hold its ground when competing against loss-based flows (e.g., loss-based TCP), it is important to distinguish between different levels of observed queuing delay. For instance, over wired connections, a moderate queuing delay value on the order of tens of milliseconds is likely self-inflicted or induced by other delay-based flows, whereas a high queuing delay value of several hundreds of milliseconds may indicate the presence of a loss-based flow that does not refrain from increased delay.

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

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

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

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

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

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

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

The aggregate congestion signal is:

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

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

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

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

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

Otherwise the algorithm operates in graduate update mode.

4.3. Sender-Side Algorithm

The sender-side algorithm is outlined as follows:

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

on receiving feedback report:
  obtain current timestamp from system clock: t_curr
  obtain values of rmode, x_curr, and r_recv from feedback report
  update estimation of rtt
  measure feedback interval: delta = t_curr - t_last
  if rmode == 0:
    update r_ref following accelerated ramp-up rules
  else:
    update r_ref following gradual update rules

  clip rate r_ref within the range of minimum rate (RMIN)
  and maximum rate (RMAX).
  x_prev = x_curr
  t_last = t_curr
```

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

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

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

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

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

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

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

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

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

At equilibrium, the aggregated congestion signal stabilizes at x_curr = PRIO*XREF*RMAX/r_ref. This ensures that when multiple flows share the same bottleneck and observe a common value of x_curr, their rates at equilibrium will be proportional to their respective priority levels (PRIO) and the range between minimum and maximum rate. Values of the minimum rate (RMIN) and maximum rate (RMAX) will be provided by the media codec, for instance, as outlined by [I-D.ietf-rmcat-cc-codec-interactions]. In the absence of such information, NADA sender will choose a default value of 0 for RMIN, and 3Mbps for RMAX.

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

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

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

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

5. Practical Implementation of NADA

5.1. Receiver-Side Operation

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

5.1.1. Estimation of one-way delay and queuing delay

The delay estimation process in NADA follows a similar approach as in earlier delay-based congestion control schemes, such as LEDBAT [RFC6817]. For experimental implementations, instead of relying on RTP timestamps and the transmission time offset RTP header extension [RFC5450], the NADA sender can generate its own timestamp based on local system clock and embed that information in the transport packet header. The NADA receiver estimates the forward delay as having a constant base delay component plus a time varying queuing delay component. The base delay is estimated as the minimum value of one-way delay observed over a relatively long period (e.g., tens of minutes), whereas the individual queuing delay value is taken to be the difference between one-way delay and base delay. By re-estimating the base delay periodically, one can avoid the potential issue of base delay expiration, whereby an earlier measured base delay value is no longer valid due to underlying route changes or cumulative timing difference introduced by the clock rate skew between sender and receiver. All delay estimations are based on sender timestamps with a recommended granularity of 100 microseconds or finer.

The individual sample values of queuing delay should be further filtered against various non-congestion-induced noise, such as spikes due to processing "hiccup" at the network nodes. Therefore, in addition to calculating the value of queuing delay using d_queue = d_fwd - d_base, as expressed in Section 5.1, current implementation further employs a minimum filter with a window size of 15 samples over per-packet queuing delay values.

5.1.2. Estimation of packet loss/marking ratio

The receiver detects packet losses via gaps in the RTP sequence numbers of received packets. For interactive real-time media application with stringent latency constraint (e.g., video conferencing), the receiver avoids the packet re-ordering delay by treating out-of-order packets as losses. The instantaneous packet

loss ratio p_inst is estimated as the ratio between the number of missing packets over the number of total transmitted packets within the recent observation window LOGWIN. The packet loss ratio p_loss is obtained after exponential smoothing:

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

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

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

5.1.3. Estimation of receiving rate

It is fairly straightforward to estimate the receiving rate r_recv. NADA maintains a recent observation window with time span of LOGWIN, and simply divides the total size of packets arriving during that window over the time span. The receiving rate (r_recv) can be calculated at either the sender side based on the per-packet feedback from the receiver, or included as part of the feedback report.

5.2. Sender-Side Operation

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

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

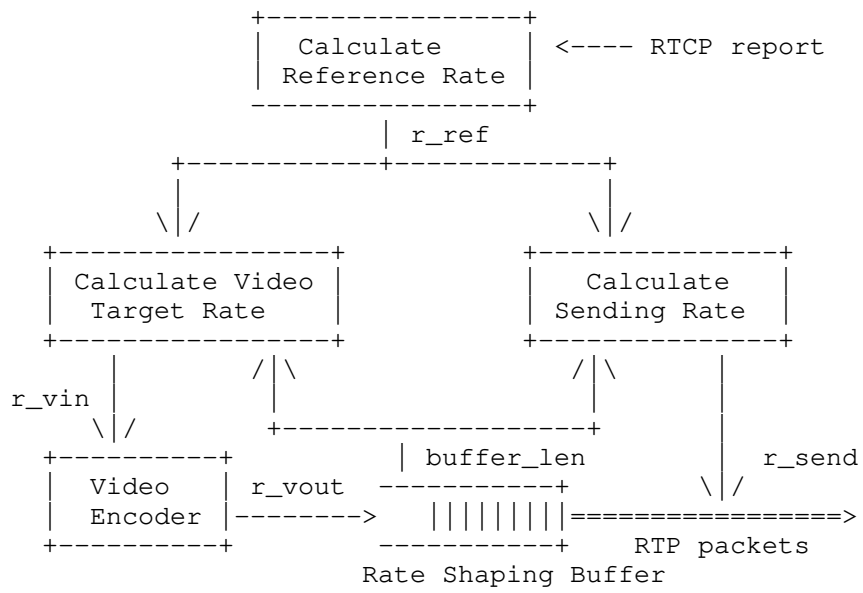


Figure 4: NADA Sender Structure

5.2.1. Rate shaping buffer

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

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

A large rate shaping buffer contributes to higher end-to-end delay, which may harm the performance of real-time media communications. Therefore, the sender has a strong incentive to prevent the rate shaping buffer from building up. The mechanisms adopted are:

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

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

5.2.2. Adjusting video target rate and sending rate

If the level of occupancy in the rate shaping buffer is accessible at the sender, such information can be leveraged to further adjust the target rate of the live video encoder r_vin as well as the actual sending rate r_send. The purpose of such adjustments is to mitigate the additional latencies introduced by the rate shaping buffer. The amount of rate adjustment can be calculated as follows:

```
r_diff_v = min(0.05*r_ref, BETA_V*8*buffer_len*FPS).     (11)
r_diff_s = min(0.05*r_ref, BETA_S*8*buffer_len*FPS).     (12)
r_vin  = max(RMIN, r_ref - r_diff_v).      (13)
r_send = min(RMAX, r_ref + r_diff_s).    (14)
```

In (11) and (12), the amount of adjustment is calculated as proportional to the size of the rate shaping buffer but is bounded by 5% of the reference rate r_ref calculated from network congestion feedback alone. This ensures that the adjustment introduced by the rate shaping buffer will not counteract with the core congestion control process. Equations (13) and (14) indicate the influence of the rate shaping buffer. A large rate shaping buffer nudges the encoder target rate slightly below -- and the sending rate slightly above -- the reference rate r_ref. The final video target rate (r_vin) and sending rate (r_send) are further bounded within the original range of [RMIN, RMAX].

Intuitively, the amount of extra rate offset needed to completely drain the rate shaping buffer within the duration of a single video frame is given by 8*buffer_len*FPS, where FPS stands for the reference frame rate of the video. The scaling parameters BETA_V and BETA_S can be tuned to balance between the competing goals of maintaining a small rate shaping buffer and deviating from the reference rate point. Empirical observations show that the rate shaping buffer for a responsive live video encoder typically stays empty and only occasionally holds a large frame (e.g., when an intra-frame is produced) in transit. Therefore, the rate adjustment introduced by this mechanism is expected to be minor. For instance, a rate shaping buffer of 2000 Bytes will lead to a rate adjustment of 48Kbps given the recommended scaling parameters of BETA_V = 0.1 and BETA_S = 0.1 and reference frame rate of FPS = 30.

5.3. Feedback Message Requirements

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

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

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

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

The above list of information can be accommodated by 48 bits, or 6 bytes, in total. They can be either included in the feedback report from the receiver, or, in the case where all receiver-side calculations are moved to the sender, derived from per-packet information from the feedback message as defined in [I-D.ietf-avtcore-cc-feedback-message]. Choice of the feedback message interval DELTA is discussed in Section 6.3. A target feedback interval of DELTA=100ms is recommended.

6. Discussions and Further Investigations

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

6.1. Choice of delay metrics

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

standing queue along the path, and sufficiently short so as to timely reflect "true" changes in minimum OWD introduced by route changes and other rare events and to mitigate the cumulative impact of clock rate skew over time.

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

Alternatively, one could move the per-packet statistical handling to the sender instead and use relative round-trip-time (RTT) in lieu of relative OWD, assuming that per-packet acknowledgments are available. The main drawback of RTT-based approach is the noise in the measured delay in the reverse direction.

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

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

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

The current recommended practice of applying minimum filter with a window size of 15 samples suffices in guarding against processing delay outliers observed in wired connections. For wireless connections with a higher packet delay variation (PDV), more sophisticated techniques on de-noising, outlier rejection, and trend analysis may be needed.

More sophisticated methods in packet loss ratio calculation, such as that adopted by [Floyd-CCR00], will likely be beneficial. These alternatives are part of the experiments this document proposes.

6.3. Impact of parameter values

In the gradual rate update mode, the parameter TAU indicates the upper bound of round-trip-time (RTT) in feedback control loop. Typically, the observed feedback interval delta is close to the target feedback interval DELTA, and the relative ratio of delta/TAU versus ETA dictates the relative strength of influence from the

aggregate congestion signal offset term (x_offset) versus its recent change (x_diff), respectively. These two terms are analogous to the integral and proportional terms in a proportional-integral (PI) controller. The recommended choice of TAU=500ms, DELTA=100ms and ETA = 2.0 corresponds to a relative ratio of 1:10 between the gains of the integral and proportional terms. Consequently, the rate adaptation is mostly driven by the change in the congestion signal with a long-term shift towards its equilibrium value driven by the offset term. Finally, the scaling parameter KAPPA determines the overall speed of the adaptation and needs to strike a balance between responsiveness and stability.

The choice of the target feedback interval DELTA needs to strike the right balance between timely feedback and low RTCP feedback message counts. A target feedback interval of DELTA=100ms is recommended, corresponding to a feedback bandwidth of 16Kbps with 200 bytes per feedback message --- approximately 1.6% overhead for a 1Mbps flow. Furthermore, both simulation studies and frequency-domain analysis in [IETF-95] have established that a feedback interval below 250ms (i.e., more frequently than 4 feedback messages per second) will not break up the feedback control loop of NADA congestion control.

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

In calculating the aggregate congestion signal x_curr, the choice of DMARK and DLOSS influence the steady-state packet loss/marking ratio experienced by the flow at a given available bandwidth. Higher values of DMARK and DLOSS result in lower steady-state loss/marking ratios, but are more susceptible to the impact of individual packet loss/marking events. While the value of DMARK and DLOSS are fixed and predetermined in the current design, this document also encourages further explorations of 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).

6.4. Sender-based vs. receiver-based calculation

In the current design, the aggregated congestion signal x_curr is calculated at the receiver, keeping the sender operation completely independent of the form of actual network congestion indications (delay, loss, or marking) in use.

Alternatively, one can shift receiver-side calculations to the sender, whereby the receiver simply reports on per-packet information via periodic feedback messages as defined in [I-D.ietf-avtcore-cc-feedback-message]. Such an approach enables interoperability amongst senders operating on different congestion control schemes, but requires slightly higher overhead in the feedback messages. See additional discussions in [I-D.ietf-avtcore-cc-feedback-message] regarding the desired format of the feedback messages and the recommended feedback intervals.

6.5. Incremental deployment

One nice property of NADA is the consistent video endpoint behavior irrespective of network node variations. This facilitates gradual, incremental adoption of the scheme.

Initially, the proposed congestion control mechanism can be implemented without any explicit support from the network, and relies solely on observed relative 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. Reference Implementations

The NADA scheme has been implemented in both [ns-2] and [ns-3] simulation platforms. The implementation in ns-2 hosts the calculations as described in Section 4.2 at the receiver side, whereas the implementation in ns-3 hosts these receiver-side calculations at the sender for the sake of interoperability. Extensive ns-2 simulation evaluations of an earlier version of the draft are documented in [Zhu-PV13]. An open source implementation of NADA as part of a ns-3 module is available at [ns3-rmcat].

Evaluation results of the current draft based on ns-3 are presented in [IETF-90] and [IETF-91] for wired test cases as documented in [I-D.ietf-rmcat-eval-test]. Evaluation results of NADA over WiFi-based test cases as defined in [I-D.ietf-rmcat-wireless-tests] are presented in [IETF-93]. These simulation-based evaluations have shown that NADA flows can obtain their fair share of bandwidth when competing against each other. They typically adapt fast in reaction to the arrival and departure of other flows, and can sustain a reasonable throughput when competing against loss-based TCP flows.

[IETF-90] describes the implementation and evaluation of NADA in a lab setting. Preliminary evaluation results of NADA in single-flow and multi-flow test scenarios have been presented in [IETF-91].

A reference implementation of NADA has been carried out by modifying the WebRTC module embedded in the Mozilla open source browser. Presentations from [IETF-103] and [IETF-105] document real-world evaluations of the modified browser driven by NADA. The experimental setting involve remote connections with endpoints over either home or enterprise wireless networks. These evaluations validate the effectiveness of NADA flows in recovering quickly from throughput drops caused by intermittent delay spikes over the last-hop wireless connections.

8. Suggested Experiments

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

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

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

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

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

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

9. IANA Considerations

This document makes no request of IANA.

10. Security Considerations

The rate adaptation mechanism in NADA relies on feedback from the receiver. As such, it is vulnerable to attacks where feedback messages are hijacked, replaced, or intentionally injected with misleading information resulting in denial of service, similar to those that can affect TCP. It is therefore RECOMMENDED that the RTCP feedback message is at least integrity checked. In addition, [I-D.ietf-avtcore-cc-feedback-message] discusses the potential risk of a receiver providing misleading congestion feedback information and the mechanisms for mitigating such risks.

The modification of sending rate based on send-side rate shaping buffer may lead to temporary excessive congestion over the network in the presence of a unresponsive video encoder. However, this effect can be mitigated by limiting the amount of rate modification introduced by the rate shaping buffer, bounding the size of the rate shaping buffer at the sender, and maintaining a maximum allowed sending rate by NADA.

11. Acknowledgments

The authors would like to thank Randell Jesup, Luca De Cicco, Piers O'Hanlon, Ingemar Johansson, Stefan Holmer, Cesar Ilharco Magalhaes, Safiqul Islam, Michael Welzl, Mirja Kuhlewind, Karen Elisabeth Egede Nielsen, Julius Flohr, Roland Bless, Andreas Smas, and Martin Stiemerling for their valuable review comments and helpful input to this specification.

12. Contributors

The following individuals have contributed to the implementation and evaluation of the proposed scheme, and therefore have helped to validate and substantially improve this specification.

Paul E. Jones <paulej@packetizer.com> of Cisco Systems implemented an early version of the NADA congestion control scheme and helped with its lab-based testbed evaluations.

Jiantao Fu <jianfu@cisco.com> of Cisco Systems helped with the implementation and extensive evaluation of NADA both in Mozilla web browsers and in earlier simulation-based evaluation efforts.

Stefano D'Aronco <stefano.daronco@geod.baug.ethz.ch> of ETH Zurich (previously at Ecole Polytechnique Federale de Lausanne when contributing to this work) helped with implementation and evaluation of an early version of NADA in [ns-3].

Charles Ganzhorn <charles.ganzhorn@gmail.com> contributed to the testbed-based evaluation of NADA during an early stage of its development.

13. References

13.1. Normative References

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

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

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

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

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

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

13.2. Informative References

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

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

[I-D.ietf-avtcore-cc-feedback-message]
Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", draft-ietf-avtcore-cc-feedback-message-04 (work in progress), July 2019.

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

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

[I-D.ietf-rmcat-eval-test]
Sarker, Z., Singh, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-ietf-rmcat-eval-test-10 (work in progress), May 2019.

[I-D.ietf-rmcat-wireless-tests]
Sarker, Z., Johansson, I., Zhu, X., Fu, J., Tan, W., and M. Ramalho, "Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks", draft-ietf-rmcat-wireless-tests-08 (work in progress), July 2019.

[IETF-103]
Zhu, X., Pan, R., Ramalho, M., Mena, S., Jones, P., Fu, J., and S. D'Aronco, "NADA Implementation in Mozilla Browser", November 2018, <https://datatracker.ietf.org/meeting/103/materials/slides-103-rmcat-nada-implementation-in-mozilla-browser-00>.

[IETF-105]
Zhu, X., Pan, R., Ramalho, M., Mena, S., Jones, P., Fu, J., and S. D'Aronco, "NADA Implementation in Mozilla Browser and Draft Update", July 2019, <https://datatracker.ietf.org/meeting/105/materials/slides-105-rmcat-nada-update-02.pdf>.

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

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

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

[IETF-95] Zhu, X., Pan, R., Ramalho, M., Mena, S., Jones, P., Fu, J., D'Aronco, S., and C. Ganzhorn, "Updates on NADA: Stability Analysis and Impact of Feedback Intervals", April 2016, <https://www.ietf.org/proceedings/95/slides/slides-95-rmcat-5.pdf>.

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

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

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

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, DOI 10.17487/RFC5450, March 2009, <https://www.rfc-editor.org/info/rfc5450>.

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

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

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

[RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017, <https://www.rfc-editor.org/info/rfc8033>.

[RFC8290] Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm", RFC 8290, DOI 10.17487/RFC8290, January 2018, <https://www.rfc-editor.org/info/rfc8290>.

[RFC8593] Zhu, X., Mena, S., and Z. Sarker, "Video Traffic Models for RTP Congestion Control Evaluations", RFC 8593, DOI 10.17487/RFC8593, May 2019, <https://www.rfc-editor.org/info/rfc8593>.

[Zhu-PV13] Zhu, X. and R. Pan, "NADA: A Unified Congestion Control Scheme for Low-Latency Interactive Video", in Proc. IEEE International Packet Video Workshop (PV'13) San Jose, CA, USA, December 2013.

Appendix A. Network Node Operations

NADA can work with different network queue management schemes and does not assume any specific network node operation. As an example, this appendix describes three variants of queue management behavior

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

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

A.1. Default behavior of drop tail queues

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

A.2. RED-based ECN marking

In this mode, the network node randomly marks the ECN field in the IP packet header following the Random Early Detection (RED) algorithm [RFC7567]. Calculation of the marking probability involves the following steps:

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

        calculate marking probability p as:

           / 0,                     if q < q_lo;
           |
           |        q_avg - q_lo
       p= <  p_max*--------------, if q_lo <= q < q_hi;
           |         q_hi - q_lo
           |
           \ p = 1,                 if q >= q_hi.
```

Here, q_lo and q_hi corresponds to the low and high thresholds of queue occupancy. The maximum marking probability is p_max.

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

A.3. Random Early Marking with Virtual Queues

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

Calculation of the marking probability involves the following steps:

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

        update token level b_tk;

        calculate the marking probability as:

         / 0,                      if b-b_tk < b_lo;
         |
         |          b-b_tk-b_lo
    p = <  p_max* --------------, if b_lo<= b-b_tk <b_hi;
         |           b_hi-b_lo
         |
         \ 1,                      if b-b_tk>=b_hi.
```

Here, the token bucket lower and upper limits are denoted by b_lo and b_hi, respectively. The parameter b indicates the size of the token bucket. The parameter r is chosen to be below capacity, resulting in slight under-utilization of the link. The maximum marking probability is p_max.

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

Authors' Addresses

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

Email: xiaoqzhu@cisco.com

Rong Pan *
* Pending affiliation change.

Email: rong.pan@gmail.com

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

Phone: +1 919 476 2038
Email: mar42@cornell.edu

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

Email: semena@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 7, 2020

C. Perkins
University of Glasgow
November 4, 2019

RTP Control Protocol (RTCP) Feedback for Congestion Control in Interactive Multimedia Conferences
draft-ietf-rmcat-rtp-cc-feedback-05

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on May 7, 2020.

Copyright Notice

Table of Contents

1. Introduction

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

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

2. Possible Models for RTCP Feedback

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

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

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

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

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

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

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

packets are sent, the less data can be included in each packet to keep the overhead constant. Does the congestion control need high rate but simple feedback (e.g., like TCP acknowledgements), or is it acceptable to send more complex feedback less often?

3. What Feedback is Achievable With RTCP?

3.1. Scenario 1: Voice Telephony

In many ways, point-to-point voice telephony is the simplest scenario for congestion control, since there is only a single media stream to control. It's complicated, however, by severe bandwidth constraints on the feedback, to keep the overhead manageable.

Assume a two-party point-to-point voice-over-IP call, using RTP over UDP/IP. A rate adaptive speech codec, such as Opus, is used, encoded into RTP packets in frames of duration Tf seconds (Tf = 20ms in many cases, but values up to 60ms are not uncommon). The congestion control algorithm requires feedback every Nr frames, i.e., every Nr * Tf seconds, to ensure effective control. Both parties in the call send speech data or comfort noise with sufficient frequency that they are counted as senders for the purpose of the RTCP reporting interval calculation.

RTCP feedback packets can be full, compound, RTCP feedback packets, or non-compound RTCP packets. A compound RTCP packet is sent once for every Nnc non-compound RTCP packets.

Compound RTCP packets contain a Sender Report (SR) packet and a Source Description (SDES) packet, and an RTP Congestion Control Feedback (CCFB) packet [I-D.ietf-avtcore-cc-feedback-message]. Non-compound RTCP packets contain only the CCFB packet. Since each participant sends only a single media stream, the extensions for RTCP report aggregation [RFC8108] and reporting group optimisation [I-D.ietf-avtcore-rtp-multi-stream-optimisation] are not used.

Within each compound RTCP packet, the SR packet will contain a sender information block (28 octets) and a single reception report block (24 octets), for a total of 52 octets. A minimal SDES packet will contain a header (4 octets) and a single chunk containing an SSRC (4 octets) and a CNAME item, and if the recommendations for choosing the CNAME [RFC7022] are followed, the CNAME item will comprise a 2 octet header, 16 octets of data, and 2 octets of padding, for a total SDES packet size of 28 octets. The CCFB packets contains an RTCP header and SSRC (8 octets), a report timestamp (4 octets), the SSRC, beginning and ending sequence numbers (8 octets), and 2*Nr octets of reports, for a total of 20 + 2*Nr octets. If IPv4 is used, with no

IP options, the UDP/IP header will be 28 octets in size. This gives a total compound RTCP packet size of Sc = 128 + 2*Nr octets.

The non-compound RTCP packets will comprise just the CCFB packet with a UDP/IP header. It can be seen that these packets will be Snc = 48 + 2*Nr octets in size.

The RTCP reporting interval calculation ([RFC3550], Section 6.2) for a two-party session where both participants are senders, reduces to Trtcp = n * Srtcp/Brtcp where Srtcp = (Sc + Nnc * Snc)/(1 + Nnc) is the average RTCP packet size in octets, Brtcp is the bandwidth allocated to RTCP in octets per second, and n is the number of participants (n=2 in this scenario).

To ensure a report is sent every Nr frames, it is necessary to set the RTCP reporting interval Trtcp = Nr * Tf, which when substituted into the previous gives Nr * Tf = n * Srtcp/Brtcp.

Solving for the RTCP bandwidth, Brtcp, and expanding the definition of Srtcp gives Brtcp = (n * (Sc + Nnc * Snc))/(Nr * Tf * (1 + Nnc)).

If we assume every report is a compound RTCP packet (i.e., Nnc = 0), the frame duration Tf = 20ms, and an RTCP report is sent for every second frame (i.e., 25 RTCP reports per second), this expression gives the needed RTCP bandwidth Brtcp = 51.6kbps. Increasing the frame duration, or reducing the frequency of reports, reduces the RTCP bandwidth, as shown below:

Tf (seconds)	Nr (frames)	rtcp_bw (kbps)
20ms	2	51.6
20ms	4	26.6
20ms	8	14.1
20ms	16	7.8
60ms	2	17.2
60ms	4	8.9
60ms	8	4.7
60ms	16	2.6

Table 1: Required RTCP bandwidth for VoIP feedback

The final row of the table (60ms frames, report every 16 frames) sends RTCP reports once per second, giving an RTCP bandwidth of 2.6kbps.

The overhead can be reduced by sending some reports in non-compound RTCP packets [RFC5506]. For example, if we alternate compound and non-compound RTCP packets, i.e., Nnc = 1, the calculation gives:

Tf (seconds)	Nr (frames)	rtcp_bw (kbps)
20ms	2	35.9
20ms	4	18.8
20ms	8	10.2
20ms	16	5.9
60ms	2	12.0
60ms	4	6.2
60ms	8	3.4
60ms	16	2.0

Table 2: Required RTCP bandwidth for VoIP feedback (alternating compound and non-compound reports)

The RTCP bandwidth needed for 60ms frames, reporting every 16 frames (once per second), can be seen to drop to 2.0kbps. This calculation can be repeated for other patterns of compound and non-compound RTCP packets, feedback frequency, and frame duration, as needed.

Note: To achieve the RTCP transmission intervals above the RTP/SAVPF profile with T_rr_interval=0 is used, since even when using the reduced minimal transmission interval, the RTP/SAVP profile would only allow sending RTCP at most every 0.11s (every third frame of video). Using RTP/SAVPF with T_rr_interval=0 however is capable of fully utilizing the configured 5% RTCP bandwidth fraction.

3.2. Scenario 2: Point-to-Point Video Conference

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

Assume all four flows are sent in a single RTP session, each using a separate SSRC; the RTCP reports from co-located audio and video SSRCs at each end point are aggregated [RFC8108]; the optimisations in [I-D.ietf-avtcore-rtp-multi-stream-optimisation] are used; and congestion control feedback is sent [I-D.ietf-avtcore-cc-feedback-message].

When all members are senders, the RTCP timing rules in Section 6.2 and 6.3 of [RFC3550] and [RFC4585] reduce to:

rtcp_interval = avg_rtcp_size * n / rtcp_bw

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

The average RTCP size will depend on the amount of feedback that is sent in each RTCP packet, on the number of members in the session, on the size of source description (RTCP SDES) information sent, and on the amount of congestion control feedback sent in each packet.

As a baseline, each RTCP packet will be a compound RTCP packet that contains an aggregate of a compound RTCP packet generated by the video SSRC and a compound RTCP packet generated by the audio SSRC. Since the RTCP reporting group extensions are used, one of these SSRCs will be a reporting SSRC, and the other will delegate its reports to that.

The aggregated compound RTCP packet from the non-reporting SSRC will contain an RTCP SR packet, an RTCP SDES packet, and an RTCP RGRS packet. The RTCP SR packet contains the 28 octet header and sender information, but no report blocks (since the reporting is delegated). The RTCP SDES packet will comprise a header (4 octets), originating SSRC (4 octets), a CNAME chunk, a terminating chunk, and any padding. If the CNAME follows [RFC7022] and [I-D.ietf-rtcweb-rtp-usage] it will be 18 octets in size, and will need 1 octet of padding, making the SDES packet 28 octets in size. The RTCP RGRS packet will be 12 octets in size. This gives a total of 28 + 28 + 12 = 68 octets.

The aggregated compound RTCP packet from the reporting SSRC will contain an RTCP SR packet, an RTCP SDES packet, and an RTCP congestion control feedback packet. The RTCP SR packet will contain two report blocks, one for each of the remote SSRCs (the report for the other local SSRC is suppressed by the reporting group extension), for a total of 28 + (2 * 24) = 76 octets. The RTCP SDES packet will comprise a header (4 octets), originating SSRC (4 octets), a CNAME chunk, an RGRP chunk, a terminating chunk, and any padding. If the CNAME follows [RFC7022] and [I-D.ietf-rtcweb-rtp-usage] it will be 18 octets in size. The RGRP chunk similarly comprises 18 octets, and 3 octets of padding are needed, for a total of 48 octets. The RTCP congestion control feedback (CCFB) report comprises an 8 octet RTCP header and SSRC,an 4 report timestamp, then for each of the remote audio and video SSRCs, an 8 octet report header, and 2 octets per packet reported upon, and padding to a 4 octet boundary if needed;

that is 8 + 4 + 8 + (2 * Nv) + 8 + (2 * Na) where Nv is the number of video packets per report, and Na is the number of audio packets per report.

The complete compound RTCP packet contains the RTCP packets from both the reporting and non-reporting SSRCs, an SRTP authentication tag, and a UDP/IPv4 header. The size of this RTCP packet is therefore: 248 + (2 * Nv) + (2 * Na) octets. Since the aggregate RTCP packet contains reports from two SSRCs, the RTCP packet size is halved before use [RFC8108]. Accordingly, we define Sc = (248 + (2 * Nv) + (2 * Na))/2 for this scenario.

How many packets does the RTCP XR congestion control feedback packet report on? This is obviously highly dependent on the choice of codec and encoding parameters, and might be quite bursty if the codec sends I-frames from which later frames are predicted. For now though, assume constant rate media with an MTU around 1500 octets, with reports for both audio and video being aggregated and sent to align with video frames. This gives the following, assuming Nr =1 and Nnc = 0 (i.e., send a compound RTCP packet for each video frame, and no non-compound packets), and using the calculation from Scenario 1: Brtcp = (n * (Sc + Nnc * Snc))/(Nr * Tf * (1 + Nnc))

Data Rate (kbps)	Video Frame Rate	Video Packets per Report: Nv	Audio Packets per Report: Na	Required RTCP bandwidth: Brtcp (kbps)
100	8	1	6	32.8 (32%)
200	16	1	3	64.0 (32%)
350	30	1	2	119.1 (34%)
700	30	2	2	120.0 (17%)
700	60	1	1	236.2 (33%)
1024	30	3	2	120.9 (11%)
1400	60	2	1	238.1 (17%)
2048	30	6	2	123.8 (6%)
2048	60	3	1	240.0 (11%)
4096	30	12	2	129.4 (3%)
4096	60	6	1	245.6 (5%)

Table 3: Required RTCP bandwidth, reporting on every frame

The RTCP bandwidth needed scales inversely with Nr. That is, it is halved if Nr=2 (report on every second packet), is reduced to one-third if Nr=3 (report on every third packet), and so on.

The needed RTCP bandwidth scales as a percentage of the data rate following the ratio of the frame rate to the data rate. As can be seen from the table above, the RTCP bandwidth needed is a significant fraction of the media rate, if reporting on every frame for low rate video. This can be solved by reporting less often at lower rates. For example, to report on every frame of 100kbps/8fps video requires the RTCP bandwidth to be 21% of the media rate; reporting every fourth frame (i.e., twice per second) reduces this overhead to 5%.

Use of reduced size RTCP [RFC5506] would allow the SR and SDES packets to be omitted from some reports. These "non-compound" (actually, compound but reduced size in this case) RTCP packets would contain an RTCP RGRS packet from the non-reporting SSRC, and an RTCP SDES RGRP packet and a congestion control feedback packet from the reporting SSRC. This will be 12 + 28 + 12 + 8 + 2*Nv + 8 + 2*Na octets, plus UDP/IP header. That is, Snc = (96 + 2*Nv + 2*Na)/2. Repeating the analysis above, but alternating compound and non-compound reports, i.e., setting Nnc = 1, gives:

Data Rate (kbps)	Video Frame Rate	Video Packets per Report: Nv	Audio Packets per Report: Na	Required RTCP bandwidth: Brtcp (kbps)
100	8	1	6	23.2 (23%)
200	16	1	3	45.0 (22%)
350	30	1	2	83.4 (23%)
700	30	2	2	84.4 (12%)
700	60	1	1	165.0 (23%)
1024	30	3	2	85.3 (8%)
1400	60	2	1	166.9 (11%)
2048	30	6	2	88.1 (4%)
2048	60	3	1	168.8 (8%)
4096	30	12	2	93.8 (2%)
4096	60	6	1	174.4 (4%)

Table 4: Required RTCP bandwidth, reporting on every frame, with reduced-size reports

The use of reduced-size RTCP gives a noticeable reduction in the needed RTCP bandwidth, and can be combined with reporting every few frames rather than every frames. Overall, it is clear that the RTCP overhead can be reasonable across the range of data and frame rates, if RTCP is configured carefully.

3.3. Scenario 3: Group Video Conference

(tbd)

3.4. Scenario 4: Screen Sharing

(tbd)

4. Discussion and Conclusions

RTCP as it is currently specified cannot be used to send per-packet congestion feedback. RTCP can, however, be used to send congestion feedback on each frame of video sent, provided the session bandwidth exceeds a couple of megabits per second (the exact rate depending on the number of session participants, the RTCP bandwidth fraction, and what RTCP extensions are enabled, and how much detail of feedback is needed). For lower rate sessions, the overhead of reporting on every frame becomes high, but can be reduced to something reasonable by sending reports once per N frames (e.g., every second frame), or by sending non-compound RTCP reports in between the regular reports.

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

The format described in [I-D.ietf-avtcore-cc-feedback-message] seems sufficient for the needs of congestion control feedback. There is little point optimising this format: the main overhead comes from the UDP/IP headers and the other RTCP packets included in the compound packets, and can be lowered by using the [RFC5506] extensions and sending reports less frequently.

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

5. Security Considerations

An attacker that can modify or spoof RTCP congestion control feedback packets can manipulate the sender behaviour to cause denial of service. This can be prevented by authentication and integrity

protection of RTCP packets, for example using the secure RTP profile [RFC3711][RFC5124], or by other means as discussed in [RFC7201].

6. IANA Considerations

There are no actions for IANA.

7. Acknowledgements

Thanks to Magnus Westerlund and the members of the RMCAT feedback design team for their feedback.

8. Informative References

[I-D.ietf-avtcore-cc-feedback-message]
Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", draft-ietf-avtcore-cc-feedback-message-04 (work in progress), July 2019.

[I-D.ietf-avtcore-rtp-multi-stream-optimisation]
Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback", draft-ietf-avtcore-rtp-multi-stream-optimisation-12 (work in progress), March 2016.

[I-D.ietf-rtcweb-rtp-usage]
Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-26 (work in progress), March 2016.

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

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/rfc4585>.

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

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <https://www.rfc-editor.org/info/rfc5506>.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, DOI 10.17487/RFC7022, September 2013, <https://www.rfc-editor.org/info/rfc7022>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <https://www.rfc-editor.org/info/rfc7201>.

[RFC8108] Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", RFC 8108, DOI 10.17487/RFC8108, March 2017, <https://www.rfc-editor.org/info/rfc8108>.

Author's Address

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

Email: csp@csperkins.org

RTP Media Congestion Avoidance Techniques D. Hayes, Ed.
Internet-Draft University of Oslo
Intended status: Experimental S. Ferlin
Expires: September 22, 2016 Simula Research Laboratory
M. Welzl
K. Hiorth
University of Oslo
March 21, 2016

Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.
draft-ietf-rmcat-sbd-04

Abstract

This document describes a mechanism to detect whether end-to-end data flows share a common bottleneck. It relies on summary statistics that are calculated by a data receiver based on continuous measurements and regularly fed to a grouping algorithm that runs wherever the knowledge is needed. This mechanism complements the coupled congestion control mechanism in draft-ietf-rmcat-coupled-cc.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 22, 2016.

Copyright Notice

Table of Contents

1. Introduction

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

1.1. The signals

The current Internet is unable to explicitly inform endpoints as to which flows share bottlenecks, so endpoints need to infer this from whatever information is available to them. The mechanism described here currently utilises packet loss and packet delay, but is not restricted to these.

1.1.1. Packet Loss

Packet loss is often a relatively rare signal. Therefore, on its own it is of limited use for SBD, however, it is a valuable supplementary measure when it is more prevalent.

1.1.2. Packet Delay

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

Measuring absolute OWD is difficult since it requires both the sender and receiver clocks to be synchronised. However, since the statistics being collected are relative to the mean OWD, a relative

OWD measurement is sufficient. Clock skew is not usually significant over the time intervals used by this SBD mechanism (see [RFC6817] A.2 for a discussion on clock skew and OWD measurements). However, in circumstances where it is significant, Section 3.4.2 outlines a way of adjusting the calculations to cater for it.

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

1.1.3. Path Lag

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

2. Definitions

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

Acronyms used in this document:

OWD -- One Way Delay

MAD -- Mean Absolute Deviation

RTT -- Round Trip Time

SBD -- Shared Bottleneck Detection

Conventions used in this document:

T -- the base time interval over which measurements are made.

N -- the number of base time, T, intervals used in some calculations.

M -- the number of base time, T, intervals used in some calculations.

sum_T(...) -- summation of all the measurements of the variable in parentheses taken over the interval T

sum(...) -- summation of terms of the variable in parentheses

sum_N(...) -- summation of N terms of the variable in parentheses

sum_NT(...) -- summation of all measurements taken over the interval N*T

E_T(...) -- the expectation or mean of the measurements of the variable in parentheses over T

E_N(...) -- the expectation or mean of the last N values of the variable in parentheses

E_M(...) -- the expectation or mean of the last M values of the variable in parentheses, where M <= N.

max_T(...) -- the maximum recorded measurement of the variable in parentheses taken over the interval T

min_T(...) -- the minimum recorded measurement of the variable in parentheses taken over the interval T

num_T(...) -- the count of measurements of the variable in parentheses taken in the interval T

num_VM(...) -- the count of valid values of the variable in parentheses given M records

PB -- a boolean variable indicating the particular flow was identified transiting a bottleneck in the previous interval T (i.e. Previously Bottleneck)

skew_est -- a measure of skewness in a OWD distribution.

skew_base_T -- a variable used as an intermediate step in calculating skew_est.

var_est -- a measure of variability in OWD measurements.

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

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

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

M and F -- number of values related to N

.

2.1. Parameters and their Effect

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

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

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

c_s — c_s is the threshold in skew_est used for determining whether a flow is transiting a bottleneck or not. It should be slightly negative so that a very lightly loaded path does not give a false indication. Setting c_s more negative makes the SBD mechanism less sensitive to transient and slight bottlenecks.

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

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

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

2.2. Recommended Parameter Values

Reference [Hayes-LCN14] uses T=350ms, N=50, p_l=0.1. The other parameters have been tightened to reflect minor enhancements to the algorithm outlined in Section 3.4: c_s=-0.01, p_f=p_d=0.1, p_s=0.15, p_mad=0.1, p_v=0.7. M=30, F=20, and c_h = 0.3 are additional parameters defined in the document. These are values that seem to work well over a wide range of practical Internet conditions.

3. Mechanism

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

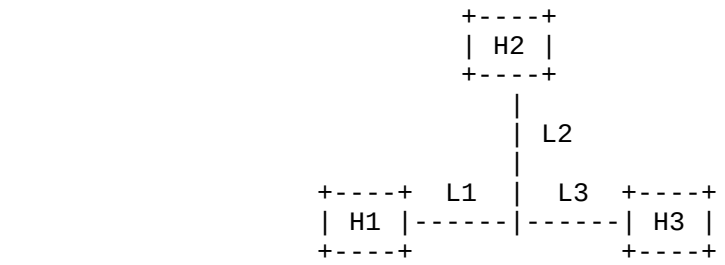
variability (estimate var_est, see Section 3.2.3)

skewness (estimate skew_est, see Section 3.2.2)

oscillation (estimate freq_est, see Section 3.2.4)

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

Summary statistics help to address both the noise and the path lag problems by describing the general shape over a relatively long period of time. Each summary statistic portrays a "view" of the bottleneck link characteristics, and when used together, they provide a robust discrimination for grouping flows. They can be signalled from a receiver, which measures the OWD and calculates the summary statistics, to a sender, which is the entity that is transmitting the media stream. An RTP Media device may be both a sender and a receiver. SBD can be performed at either a sender or a receiver or both.



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

Figure 1

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

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

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

3.1. SBD feedback requirements

There are three possible scenarios each with different feedback requirements:

1. Both summary statistic calculations and SBD are performed at senders only.

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

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

3.1.1. Feedback when all the logic is placed at the sender

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

The sender requires precise accurate OWD measurements for every packet, along with the proportion of packets lost over the interval T, to be sent from the receivers to the senders every T.

An initialisation message may be required to agree on the feedback interval.

3.1.2. Feedback when the statistics are calculated at the receiver and SBD at the sender

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

- An initialization message from the sender to the receiver will contain the following information:
  - A protocol identifier (SBD=01). This is to future proof the message exchange so that potential advances in SBD technology can be easily deployed. All following initialisation elements relate to the mechanism outlined in this document which will have the identifier SBD=01.
  - A list of which key metrics should be collected and relayed back to the sender out of a possibly extensible set (pkt_loss, var_est, skew_est, freq_est). The grouping algorithm described in this document requires all four of these metrics, and receivers MUST be able to provide them, but future algorithms may be able to exploit other metrics (e.g. metrics based on explicit network signals).
  - The values of T, N, M, and the necessary resolution and precision of the relayed statistics.
- A response message from the receiver acknowledges this message with a list of key metrics it supports (subset of the senders list) and is able to relay back to the sender.

This initialisation exchange may be repeated to finalize the agreed metrics should not all be supported by all receivers.

After initialisation the agreed summary statistics will be fed back to the sender every T.

3.1.3. Feedback when bottlenecks can be determined at both senders and receivers

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

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

After initialisation both summary statistics and shared bottleneck determinations will need to be exchanged every T.

3.2. Key metrics and their calculation

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

3.2.1. Mean delay

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

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

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

where M <= N. Setting M to be less than N allows the mechanism to be more responsive to changes, but potentially at the expense of a higher error rate (see Section 3.5 for a discussion on improving the responsiveness of the mechanism.)

3.2.2. Skewness Estimate

Skewness is difficult to calculate efficiently and accurately. Ideally it should be calculated over the entire period (M * T) from the mean OWD over that period. However this would require storing every delay measurement over the period. Instead, an estimate is

made over M * T based on a calculation every T using the previous T's calculation of mean_delay.

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

if (OWD < mean_delay) skew_base_T++

if (OWD > mean_delay) skew_base_T--

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

skew_est = sum_MT(skew_base_T)/num_MT(OWD)

where skew_est is a number between -1 and 1

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

3.2.3. Variability Estimate

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

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

where

|x| is the absolute value of x

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

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

For calculation of freq_est p_v=0.7

For the grouping threshold p_mad=0.1

3.2.4. Oscillation Estimate

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

freq_est = number_of_crossings / N

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

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

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

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

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

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

3.2.5. Packet loss

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

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

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

3.3. Flow Grouping

3.3.1. Flow Grouping Algorithm

The following grouping algorithm is RECOMMENDED for SBD in the RMCAT context and is sufficient and efficient for small to moderate numbers of flows. For very large numbers of flows (e.g. hundreds), a more complex clustering algorithm may be substituted.

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

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

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

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

   skew_est < c_s

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

The parameter c_s controls how sensitive the mechanism is in detecting a bottleneck. C_s = 0.0 was used in [Hayes-LCN14]. A value of c_s = 0.05 is a little more sensitive, and c_s = -0.05 is a little less sensitive. C_h controls the hysteresis on flows that were grouped as transiting a bottleneck last time. If the test result is TRUE, PB=TRUE, otherwise PB=FALSE.

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

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

   diff(freq_est) < p_f

3. Group flows whose difference in sorted E_M(var_est) (highest to lowest) is less than a threshold:

   diff(var_est) < (p_mad * var_est)

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

4. Group flows whose difference in sorted skew_est is less than a threshold:

   diff(skew_est) < p_s

5. When packet loss is high enough to be reliable (pkt_loss > p_l), group flows whose difference is less than a threshold

   diff(pkt_loss) < (p_d * pkt_loss)

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

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

3.3.2. Using the flow group signal

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

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

3.4. Removing Noise from the Estimates

The following describe small changes to the calculation of the key metrics that help remove noise from them. Currently these "tweaks" are described separately to keep the main description succinct. In future revisions of the draft these enhancements may replace the original key metric calculations.

3.4.1. Oscillation noise

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

To remove this source of noise from freq_est:

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

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

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

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

3.4.2. Clock skew

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

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

3.5. Reducing lag and Improving Responsiveness

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

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

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

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

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

3.5.1. Improving the response of the skewness estimate

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

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

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

To calculate this weighted skew_est incrementally:

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

Initialise: sum_skewbase = 0, F_skewbase=0, W_D_skewbase=0

skewbase_hist = buffer length M initialize to 0

numsampT = buffer length M initialzed to 0

Steps per iteration:

1. old_skewbase = skewbase_hist(M)

2. old_numsampT = numsampT(M)

3. cycle(skewbase_hist)

4. cycle(numsampT)

5. numsampT(1) = num_T(OWD)

6. skewbase_hist(1) = skew_base_T

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

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

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

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

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

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

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

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

3.5.2. Improving the response of the variability estimate

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

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

where numsampT is an array of the number of OWD samples in each T (i.e. num_T(OWD)), and numsampT(1) is the most recent; skew_base_T(1) is the most recent calculation of skew_base_T; 1:F refers to the integer values 1 through to F, and [(M-F):1] refers to an array of the integer values (M-F) declining through to 1; and ".*" is the array scalar dot product operator. When removing oscillation noise (see Section 3.4.1) this calculation must be adjusted to allow for invalid var_base_T records.

Var_est can be calculated incrementally in the same way as skew_est in Section 3.5.1. However, note that the buffer numsampT is used for both calculations so the operations on it should not be repeated.

4. Measuring OWD

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

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

4.1. Time stamp resolution

The SBD mechanism requires timing information precise enough to be able to make comparisons. As a rule of thumb, the time resolution should be less than one hundredth of a typical path’s range of delays. In general, the lower the time resolution, the more care that needs to be taken to ensure rounding errors do not bias the skewness calculation.

Typical RTP media flows use sub-millisecond timers, which should be adequate in most situations.

5. Implementation status

The University of Oslo is currently working on an implementation of this in the Chromium browser.

6. Acknowledgements

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

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

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

Non-authenticated RTCP packets carrying shared bottleneck indications and summary statistics could allow attackers to alter the bottleneck sharing characteristics for private gain or disruption of other parties communication.

9. Change history

Changes made to this document:

WG-03->WG-04 :	Add M to terminology table, suggest skew_est based on previous T and no freq_est in clock skew section, feedback requirements as a separate sub section.
WG-02->WG-03 :	Correct misspelled author
WG-01->WG-02 :	Removed ambiguity associated with the term "congestion". Expanded the description of initialisation messages. Removed PDV metric. Added description of incremental weighted metric calculations for skew_est. Various clarifications based on implementation work. Fixed typos and tuned parameters.

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

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

01->02 : New section describing improvements to the key metric calculations that help to remove noise, bias, and reduce lag. Some revisions to the notation to make it clearer. Some tightening of the thresholds.

00->01 : Revisions to terminology for clarity

10. References

10.1. Normative References

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

10.2. Informative References

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

[I-D.ietf-rmcat-coupled-cc] Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-00 (work in progress), September 2015.

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

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

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

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

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

Authors' Addresses

David Hayes (editor)
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Phone: +47 2284 5566
Email: davihay@ifi.uio.no

Simone Ferlin
Simula Research Laboratory
P.O.Box 134
Lysaker 1325
Norway

Phone: +47 4072 0702
Email: ferlin@simula.no

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

Phone: +47 2285 2420
Email: michawe@ifi.uio.no

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

Email: kristahi@ifi.uio.no

RMCAT WG
Internet-Draft
Intended status: Experimental
Expires: April 29, 2018

I. Johansson
Z. Sarker
Ericsson AB

October 26, 2017

Self-Clocked Rate Adaptation for Multimedia
draft-ietf-rmcat-scream-cc-13

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 29, 2018.

Table of Contents

1. Introduction

Congestion in the Internet occurs when the transmitted bitrate is higher than the available capacity over a given transmission path. Applications that are deployed in the Internet have to employ congestion control, to achieve robust performance and to avoid congestion collapse in the Internet. Interactive realtime communication imposes a lot of requirements on the transport, therefore a robust, efficient rate adaptation for all access types is an important part of interactive realtime communications as the transmission channel bandwidth can vary over time. Wireless access such as LTE, which is an integral part of the current Internet, increases the importance of rate adaptation as the channel bandwidth of a default LTE bearer [QoS-3GPP] can change considerably in a very short time frame. Thus a rate adaptation solution for interactive realtime media, such as WebRTC, should be both quick and be able to operate over a large range in channel capacity. This memo describes SCReAM (Self-Clocked Rate Adaptation for Multimedia), a solution that implements congestion control for RTP streams [RFC3550]. While SCReAM was originally devised for WebRTC (Web Real-Time Communication) [RFC7478], it can also be used for other applications where congestion control of RTP streams is necessary. SCReAM is based on the self-clocking principle of TCP and uses techniques similar to what is used in the LEDBAT based rate adaptation algorithm [RFC6817]. SCReAM is not entirely self-clocked as it augments self-clocking with pacing and a minimum send rate.
SCReAM can take advantage of ECN (Explicit Congestion Notification) in cases where ECN is supported by the network and the hosts. ECN is however not required for the basic congestion control functionality in SCReAM.

1.1. Wireless (LTE) access properties

[I-D.ietf-rmcat-wireless-tests] describes the complications that can be observed in wireless environments. Wireless access such as LTE can typically not guarantee a given bandwidth, this is true especially for default bearers. The network throughput can vary considerably for instance in cases where the wireless terminal is moving around. Even though LTE can support bitrates well above 100Mbps, there are cases when the available bitrate can be much lower, examples are situations with high network load and poor coverage. An additional complication is that the network throughput can drop for short time intervals at e.g. handover, these short glitches are initially very difficult to distinguish from more permanent reductions in throughput.

Unlike wireline bottlenecks with large statistical multiplexing it is not possible to try to maintain a given bitrate when congestion is

detected with the hope that other flows will yield, this is because there are generally few other flows competing for the same bottleneck. Each user gets its own variable throughput bottleneck, where the throughput depends on factors like channel quality, network load and historical throughput. The bottom line is, if the throughput drops, the sender has no other option than to reduce the bitrate. Once the radio scheduler has reduced the resource allocation for a bearer, an RMCAT flow in that bearer aims to reduce the sending rate quite quickly (within one RTT) in order to avoid excessive queuing delay or packet loss.

1.2. Why is it a self-clocked algorithm?

Self-clocked congestion control algorithms provide a benefit over the rate based counterparts in that the former consists of two adaptation mechanisms:

- A congestion window computation that evolves over a longer timescale (several RTTs) especially when the congestion window evolution is dictated by estimated delay (to minimize vulnerability to e.g. short term delay variations).

- A fine grained congestion control given by the self-clocking which operates on a shorter time scale (1 RTT). The benefits of self-clocking are also elaborated upon in [TFWC].

A rate based congestion control typically adjusts the rate based on delay and loss. The congestion detection needs to be done with a certain time lag to avoid over-reaction to spurious congestion events such as delay spikes. Despite the fact that there are two or more congestion indications, the outcome is still that there is still only one mechanism to adjust the sending rate. This makes it difficult to reach the goals of high throughput and prompt reaction to congestion.

2. Terminology

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

3. Overview of SCReAM Algorithm

The core SCReAM algorithm has similarities to the concepts of self-clocking used in TFWC [TFWC] and follows the packet conservation principle. The packet conservation principle is described as an important key-factor behind the protection of networks from congestion [Packet-conservation].

In SCReAM, the receiver of the media echoes a list of received RTP packets and the timestamp of the RTP packet with the highest sequence number back to the sender in feedback packets. The sender keeps a list of transmitted packets, their respective sizes and the time they were transmitted. This information is used to determine the number of bytes that can be transmitted at any given time instant. A congestion window puts an upper limit on how many bytes can be in flight, i.e. transmitted but not yet acknowledged.

The congestion window is determined in a way similar to LEDBAT [RFC6817]. LEDBAT is a congestion control algorithm that uses send and receive timestamps to estimate the queuing delay (from now on denoted qdelay) along the transmission path. This information is used to adjust the congestion window. The use of LEDBAT ensures that the end-to-end latency is kept low. [LEDBAT-delay-impact] shows that LEDBAT has certain inherent issues that makes it counteract its purpose to achieve low delay. The general problem described in the paper is that the base delay is offset by LEDBAT's own queue buildup. The big difference with using LEDBAT in the SCReAM context lies in the fact that the source is rate limited and that it is required that the RTP queue is kept short (preferably empty). In addition the output from a video encoder is rarely constant bitrate, static content (talking heads) for instance gives almost zero video bitrate. This gives two useful properties when LEDBAT is used with SCReAM that help to avoid the issues described in [LEDBAT-delay-impact]:

1. There is always a certain probability that SCReAM is short of data to transmit, which means that the network queue will run empty every once in a while.

2. The max video bitrate can be lower than the link capacity. If the max video bitrate is 5Mbps and the capacity is 10Mbps then the network queue will run empty.

It is sufficient that any of the two conditions above is fulfilled to make the base delay update properly. Furthermore [LEDBAT-delay-impact] describes an issue with short lived competing flows, the case in SCReAM is that these short lived flows will cause the self-clocking in SCReAM to slow down with the result that the RTP queue is built up, which will in turn result in a reduced media video bitrate. SCReAM will thus yield more to competing short lived flows than what is the case with traditional use of LEDBAT.
The basic functionality in the use of LEDBAT in SCReAM is quite simple, there are however a few steps to take to make the concept work with conversational media:

- Congestion window validation techniques. These are similar in action as the method described in [RFC7661]. Congestion window

validation ensures that the congestion window is limited by the actual number bytes in flight, this is important especially in the context of rate limited sources such as video. Lack of congestion window validation would lead to a slow reaction to congestion as the congestion window does not properly reflect the congestion state in the network. The allowed idle period in this memo is shorter than in [RFC7661], this to avoid excessive delays in the cases where e.g. wireless throughput has decreased during a period where the output bitrate from the media coder has been low, for instance due to inactivity. Furthermore, this memo allows for more relaxed rules for when the congestion window is allowed to grow, this is necessary as the variable output bitrate generally means that the congestion window is often under-utilized.

- Fast increase makes the bitrate increase faster when no congestion is detected. It makes the media bitrate ramp-up within 5 to 10 seconds. The behavior is similar to TCP slowstart. The fast increase is exited when congestion is detected. The fast increase state can however resume if the congestion level is low, this enables a reasonably quick rate increase in case link throughput increases.

- A qdelay trend is computed for earlier detection of incipient congestion and as a result it reduces jitter.

- Addition of a media rate control function.

- Use of inflection points in the media rate calculation to achieve reduced jitter.

- Adjustment of qdelay target for better performance when competing with other loss based congestion controlled flows.

The above mentioned features will be described in more detail in sections Section 3.1 to Section 3.3. The full details are described in Section 4.

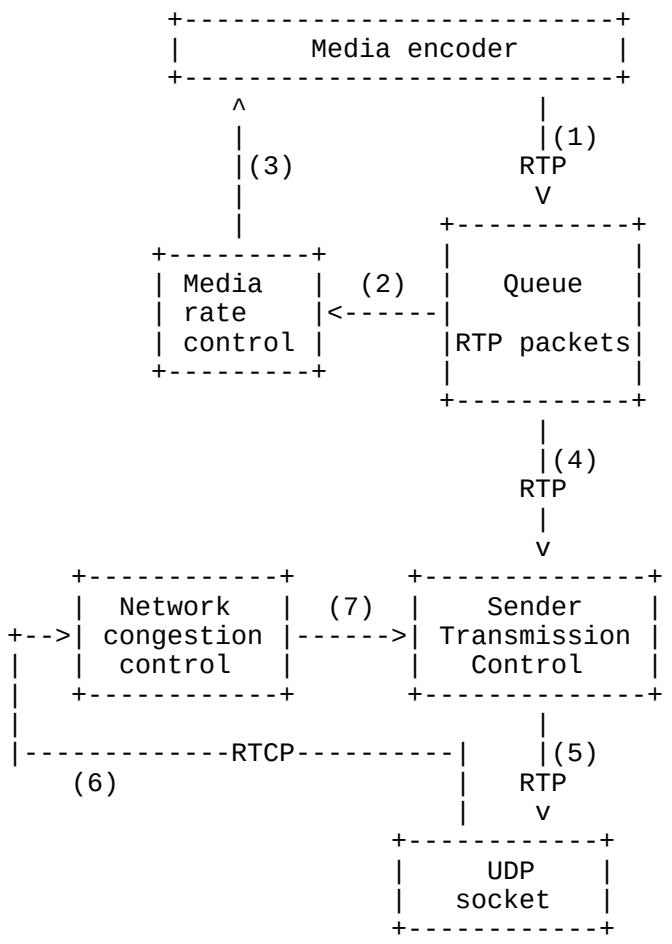


Figure 1: SCReAM sender functional view

The SCReAM algorithm consists of three main parts: network congestion control, sender transmission control and media rate control. All of these three parts reside at the sender side. Figure 1 shows the functional overview of a SCReAM sender. The receiver side algorithm is very simple in comparison as it only generates feedback containing acknowledgements of received RTP packets and an ECN count.

3.1. Network Congestion Control

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

The SCReAM congestion control method, uses techniques similar to LEDBAT [RFC6817] to measure the qdelay. As is the case with LEDBAT, it is not necessary to use synchronized clocks in sender and receiver in order to compute the qdelay. It is however necessary that they use the same clock frequency, or that the clock frequency at the receiver can be inferred reliably by the sender. Failure to meet this requirement leads to malfunction in the SCReAM congestion control algorithm due to incorrect estimation of the network queue delay.

The SCReAM sender calculates the congestion window based on the feedback from the SCReAM receiver. The congestion window is allowed to increase if the qdelay is below a predefined qdelay target, otherwise the congestion window decreases. The qdelay target is typically set to 50-100ms. This ensures that the queuing delay is kept low. The reaction to loss or ECN events leads to an instant reduction of CWND. Note that the source rate limited nature of real time media such as video, typically means that the queuing delay will mostly be below the given delay target, this is contrary to the case where large files are transmitted using LEDBAT congestion control, in which case the queuing delay will stay close to the delay target.

3.2. Sender Transmission Control

The sender transmission control limits the output of data, given by the relation between the number of bytes in flight and the congestion window. Packet pacing is used to mitigate issues with ACK compression that MAY cause increased jitter and/or packet loss in the media traffic. Packet pacing limits the packet transmission rate given by the estimated link throughput. Even if the send window allows for the transmission of a number of packets, these packets are not transmitted immediately, but rather they are transmitted in intervals given by the packet size and the estimated link throughput.

3.3. Media Rate Control

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

The reaction to reduced throughput MUST be prompt in order to avoid getting too much data queued in the RTP packet queue(s) in the sender. The media bitrate is decreased if the RTP queue size exceeds a threshold.

In cases where the sender frame queues increase rapidly such as in the case of a RAT (Radio Access Type) handover it MAY be necessary to implement additional actions, such as discarding of encoded media

frames or frame skipping in order to ensure that the RTP queues are drained quickly. Frame skipping results in the frame rate being temporarily reduced. Which method to use is a design choice and outside the scope of this algorithm description.

4. Detailed Description of SCReAM

4.1. SCReAM Sender

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

A SCReAM sender implements media rate control and an RTP queue for each media type or source, where RTP packets containing encoded media frames are temporarily stored for transmission. Figure 1 shows the details when a single media source (or stream) is used. A transmission scheduler (not shown in the figure) is added to support multiple streams. The transmission scheduler can enforce differing priorities between the streams and act like a coupled congestion controller for multiple flows. Support for multiple streams is implemented in [SCReAM-CPP-implementation].

Media frames are encoded and forwarded to the RTP queue (1) in Figure 1. The media rate adaptation adapts to the size of the RTP queue (2) and provides a target rate for the media encoder (3). The RTP packets are picked from the RTP queue (for multiple flows from each RTP queue based on some defined priority order or simply in a round robin fashion) (4) by the sender transmission controller. The sender transmission controller (in case of multiple flows a transmission scheduler) sends the RTP packets to the UDP socket (5). In the general case all media SHOULD go through the sender transmission controller and is limited so that the number of bytes in flight is less than the congestion window. RTCP packets are received (6) and the information about bytes in flight and congestion window is exchanged between the network congestion control and the sender transmission control (7).

4.1.1. Constants and Parameter values

Constants and state variables are listed in this section. Temporary variables are not listed, instead they are appended with ’_t’ in the pseudo code to indicate their local scope.

4.1.1.1. Constants

The RECOMMENDED values, within (), for the constants are deduced from experiments. The units are enclosed in square brackets [ ].

QDELAY_TARGET_LO (0.1s)
: Target value for the minimum qdelay.

QDELAY_TARGET_HI (0.4s)
: Target value for the maximum qdelay. This parameter provides an upper limit to how much the target qdelay (qdelay_target) can be increased in order to cope with competing loss based flows. The target qdelay does not have to be initialized to this high value however as it would increase e2e delay and also make the rate control and congestion control loop sluggish.

QDELAY_WEIGHT (0.1)
: Averaging factor for qdelay_fraction_avg.

QDELAY_TREND_TH (0.2)
: Threshold for the detection of incipient congestion.

MIN_CWND (3000byte)
: Minimum congestion window.

MAX_BYTES_IN_FLIGHT_HEAD_ROOM (1.1)
: Headroom for the limitation of CWND.

GAIN (1.0)
: Gain factor for congestion window adjustment.

BETA_LOSS (0.8)
: CWND scale factor due to loss event.

BETA_ECN (0.9)
: CWND scale factor due to ECN event.

BETA_R (0.9)
: Target rate scale factor due to loss event.

MSS (1000 byte)
: Maximum segment size = Max RTP packet size.

RATE_ADJUST_INTERVAL (0.2s)
: Interval between media bitrate adjustments.

TARGET_BITRATE_MIN
: Min target bitrate [bps], bps is bits per second.

TARGET_BITRATE_MAX
Max target bitrate [bps].

RAMP_UP_SPEED (200000bps/s)
Maximum allowed rate increase speed.

PRE_CONGESTION_GUARD (0.0..1.0)
Guard factor against early congestion onset. A higher value gives less jitter, possibly at the expense of a lower link utilization. This value MAY be subject to tuning depending on e.g media coder characteristics, experiments with H264 and VP8 indicate that 0.1 is a suitable value. See [SCReAM-CPP-implementation] and [SCReAM-implementation-experience] for evaluation of a real implementation.

TX_QUEUE_SIZE_FACTOR (0.0..2.0)
Guard factor against RTP queue buildup. This value MAY be subject to tuning depending on e.g media coder characteristics, experiments with H264 and VP8 indicate that 1.0 is a suitable value. See [SCReAM-CPP-implementation] and [SCReAM-implementation-experience] for evaluation of a real implementation.

RTP_QDELAY_TH (0.02s) RTP queue delay threshold for a target rate reduction.

TARGET_RATE_SCALE_RTP_QDELAY (0.95) Target rate scale when RTP qdelay threshold exceeds RTP_QDELAY_TH.

QDELAY_TREND_LO (0.2) Threshold value for qdelay_trend.

T_RESUME_FAST_INCREASE (5s) Time span until fast increase can be resumed, given that the qdelay_trend is below QDELAY_TREND_LO.

RATE_PACE_MIN (50000bps) Minimum pacing rate.

4.1.1.2. State variables

The values within () indicate initial values.

qdelay_target (QDELAY_TARGET_LO)
qdelay target, a variable qdelay target is introduced to manage cases where e.g. FTP competes for the bandwidth over the same bottleneck, a fixed qdelay target would otherwise starve the RMCAT flow under such circumstances. The qdelay target is allowed to vary between QDELAY_TARGET_LO and QDELAY_TARGET_HI.

qdelay_fraction_avg (0.0)

EWMA (Exponentially Weighted Moving Average) filtered fractional qdelay.

qdelay_fraction_hist[20] ({0,..,0})
Vector of the last 20 fractional qdelay samples.

qdelay_trend (0.0)
qdelay trend, indicates incipient congestion.

qdelay_trend_mem (0.0)
Low pass filtered version of qdelay_trend.

qdelay_norm_hist[100] ({0,..,0})
Vector of the last 100 normalized qdelay samples.

in_fast_increase (true)
True if in fast increase state.

cwnd (MIN_CWND)
Congestion window.

bytes_newly_acked (0)
The number of bytes that was acknowledged with the last received acknowledgement i.e. bytes acknowledged since the last CWND update.

max_bytes_in_flight (0)
The maximum number of bytes in flight over a sliding time window, i.e. transmitted but not yet acknowledged bytes.

send_wnd (0)
Upper limit to how many bytes that can currently be transmitted. Updated when cwnd is updated and when RTP packet is transmitted.

target_bitrate (0 bps)
Media target bitrate.

target_bitrate_last_max (1 bps)
Media target bitrate inflection point i.e. the last known highest target_bitrate. Used to limit bitrate increase speed close to the last known congestion point.

rate_transmit (0.0 bps)
Measured transmit bitrate.

rate_ack (0.0 bps)
Measured throughput based on received acknowledgements.

rate_media (0.0 bps)

Measured bitrate from the media encoder.

rate_media_median (0.0 bps)
Median value of rate_media, computed over more than 10s.

s_rtt (0.0s)
Smoothed RTT [s], computed with a similar method to that described in [RFC6298].

rtp_queue_size (0 bits)
Sum of the sizes of RTP packets in queue.

rtp_size (0 byte)
Size of the last transmitted RTP packet.

loss_event_rate (0.0)
The estimated fraction of RTTs with lost packets detected.

4.1.2. Network congestion control

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

- Computation of congestion window at the sender: Gives an upper limit to the number of bytes in flight.
- Calculation of send window at the sender: RTP packets are transmitted if allowed by the relation between the number of bytes in flight and the congestion window. This is controlled by the send window.

SCReAM is a window based and byte oriented congestion control protocol, where the number of bytes transmitted is inferred from the size of the transmitted RTP packets. Thus a list of transmitted RTP packets and their respective transmission times (wall-clock time) MUST be kept for further calculation.

The number of bytes in flight (bytes_in_flight) is computed as the sum of the sizes of the RTP packets ranging from the RTP packet most recently transmitted down to but not including the acknowledged packet with the highest sequence number. This can be translated to the difference between the highest transmitted byte sequence number and the highest acknowledged byte sequence number. As an example: If RTP packet with sequence number SN is transmitted and the last acknowledgement indicates SN-5 as the highest received sequence number then bytes in flight is computed as the sum of the size of RTP packets with sequence number SN-4, SN-3, SN-2, SN-1 and SN, it does not matter if for instance packet with sequence number SN-3 was lost,

the size of RTP packet with sequence number SN-3 will still be considered in the computation of bytes_in_flight.

Furthermore, a variable bytes_newly_acked is incremented with a value corresponding to how much the highest sequence number has increased since the last feedback. As an example: If the previous acknowledgement indicated the highest sequence number N and the new acknowledgement indicated N+3, then bytes_newly_acked is incremented by a value equal to the sum of the sizes of RTP packets with sequence number N+1, N+2 and N+3. Packets that are lost are also included, which means that even though e.g packet N+2 was lost, its size is still included in the update of bytes_newly_acked. The bytes_newly_acked variable is reset to zero after a CWND update.

The feedback from the receiver is assumed to consist of the following elements.

- A list of received RTP packets' sequence numbers.
- The wall clock timestamp corresponding to the received RTP packet with the highest sequence number.
- Accumulated number of ECN-CE marked packets (n_ECN).

When the sender receives RTCP feedback, the qdelay is calculated as outlined in [RFC6817]. A qdelay sample is obtained for each received acknowledgement. No smoothing of the qdelay samples occur, however some smoothing occurs anyway as the computation of the CWND is a low pass filter function. A number of variables are updated as illustrated by the pseudo code below, temporary variables are appended with '_t'. As mentioned in Section 7 , calculation of the proper congestion window and media bitrate may benefit from additional optimizations for handling of very high and very low bitrates, and from additional damping to handle periodic packet bursts. Some such optimizations are implemented in [SCReAM-CPP-implementation], but they do not form part of the specification of SCReAM at this time.

```
<CODE BEGINS>
update_variables(qdelay):
  qdelay_fraction_t = qdelay/qdelay_target
  # Calculate moving average
  qdelay_fraction_avg = (1-QDELAY_WEIGHT)*qdelay_fraction_avg+
     QDELAY_WEIGHT*qdelay_fraction_t
  update_qdelay_fraction_hist(qdelay_fraction_t)
  # Compute the average of the values in qdelay_fraction_hist
  avg_t = average(qdelay_fraction_hist)
  # R is an autocorrelation function of qdelay_fraction_hist,
  #  with the mean (DC component) removed, at lag K
  # The subtraction of the scalar avg_t from
  #  qdelay_fraction_hist is performed element-wise
  a_t = R(qdelay_fraction_hist-avg_t,1)/
        R(qdelay_fraction_hist-avg_t,0)
  # Calculate qdelay trend
  qdelay_trend = min(1.0,max(0.0,a_t*qdelay_fraction_avg))
  # Calculate a 'peak-hold' qdelay_trend, this gives a memory
  #  of congestion in the past
  qdelay_trend_mem = max(0.99*qdelay_trend_mem, qdelay_trend)
<CODE ENDS>
```

The qdelay fraction is sampled every 50ms and the last 20 samples are stored in a vector (qdelay_fraction_hist). This vector is used in the computation of an qdelay trend that gives a value between 0.0 and 1.0 depending on the estimated congestion level. The prediction coefficient 'a_t' has positive values if qdelay shows an increasing or decreasing trend, thus an indication of congestion is obtained before the qdelay target is reached. As a side effect, also the case that qdelay decreases is taken as a sign of congestion, experiments have however shown that this is beneficial as varying queue delay up or down is an indication that the transmit rate is very close to the path capacity.

The autocorrelation function 'R' is defined as follows. Let x be a vector constituting N values, the biased autocorrelation function for a given lag=k for the vector x is given by.

$$R(x,k) = \sum_{n=1}^{n=N-k} x(n)*x(n+k)$$

The prediction coefficient is further multiplied with qdelay_fraction_avg to reduce sensitivity to increasing qdelay when it is very small. The 50ms sampling is a simplification that could have the effect that the same qdelay is sampled several times, this does however not pose any problem as the vector is only used to determine if the qdelay is increasing or decreasing. The

qdelay_trend is utilized in the media rate control to indicate incipient congestion and to determine when to exit from fast increase mode. qdelay_trend_mem is used to enforce a less aggressive rate increase after congestion events. The function update_qdelay_fraction_hist(..) removes the oldest element and adds the latest qdelay_fraction element to the qdelay_fraction_hist vector.

4.1.2.1. Reaction to packets loss and ECN

A loss event is indicated if one or more RTP packets are declared missing. The loss detection is described in Section 4.1.2.4. Once a loss event is detected, further detected lost RTP packets SHOULD be ignored for a full smoothed round trip time, the intention of this is to limit the congestion window decrease to at most once per round trip.
The congestion window back off due to loss events is deliberately a bit less than is the case with e.g. TCP Reno. The reason is that TCP is generally used to transmit whole files, which can be translated to an infinite source bitrate. SCReAM on the other hand has a source whose rate is limited to a value close to the available transmit rate and often below that value, the effect of this is that SCReAM has less opportunity to grab free capacity than a TCP based file transfer. To compensate for this it is RECOMMENDED to let SCReAM reduce the congestion window less than what is the case with TCP when loss events occur.

An ECN event is detected if the n_ECN counter in the feedback report has increased since the previous received feedback. Once an ECN event is detected, the n_ECN counter is ignored for a full smoothed round trip time, the intention of this is to limit the congestion window decrease to at most once per round trip. The congestion window back off due to an ECN event MAY be smaller than if a loss event occurs. This is in line with the idea outlined in [I-D.ietf-tcpm-alternativebackoff-ecn] to enable ECN marking thresholds lower than the corresponding packet drop thresholds.

4.1.2.2. Congestion window update

The update of the congestion window depends on whether loss or ECN-marking or neither occurs. The pseudo code below describes actions taken in case of the different events.

```
<CODE BEGINS>
on congestion event(qdelay):
  # Either loss or ECN mark is detected
  in_fast_increase = false
  if (is loss)
    # Loss is detected
    cwnd = max(MIN_CWND,cwnd*BETA_LOSS)
  else
    # No loss, so it is then an ECN mark
    cwnd = max(MIN_CWND,cwnd*BETA_ECN)
  end
  adjust_qdelay_target(qdelay) #compensating for competing flows
  calculate_send_window(qdelay,qdelay_target)

# When no congestion event
on acknowledgement(qdelay):
  update_bytes_newly_acked()
  update_cwnd(bytes_newly_acked)
  adjust_qdelay_target(qdelay) #compensating for competing flows
  calculate_send_window(qdelay, qdelay_target)
  check_to_resume_fast_increase()
<CODE ENDS>
```

The methods are further described in detail below.

The congestion window update is based on qdelay, except for the occurrence of loss events (one or more lost RTP packets in one RTT), or ECN events, which was described earlier.

Pseudo code for the update of the congestion window is found below.

```
<CODE BEGINS>
update_cwnd(bytes_newly_acked):
  # In fast increase ?
  if (in_fast_increase)
    if (qdelay_trend >= QDELAY_TREND_TH)
      # Incipient congestion detected, exit fast increase
      in_fast_increase = false
    else
      # No congestion yet, increase cwnd if it
      #  is sufficiently used
      # An additional slack of bytes_newly_acked is
      #  added to ensure that CWND growth occurs
      #  even when feedback is sparse
      if (bytes_in_flight*1.5+bytes_newly_acked > cwnd)
        cwnd = cwnd+bytes_newly_acked
      end
      return
    end
  end

  # Not in fast increase phase
  # off_target calculated as with LEDBAT
  off_target_t = (qdelay_target - qdelay) / qdelay_target

  gain_t = GAIN
  # Adjust congestion window
  cwnd_delta_t =
    gain_t * off_target_t * bytes_newly_acked * MSS / cwnd
  if (off_target_t > 0 &&
      bytes_in_flight*1.25+bytes_newly_acked <= cwnd)
    # No cwnd increase if window is underutilized
    # An additional slack of bytes_newly_acked is
    #  added to ensure that CWND growth occurs
    #  even when feedback is sparse
    cwnd_delta_t = 0;
  end

  # Apply delta
  cwnd += cwnd_delta_t
  # limit cwnd to the maximum number of bytes in flight
  cwnd = min(cwnd, max_bytes_in_flight*MAX_BYTES_IN_FLIGHT_HEAD_ROOM)
  cwnd = max(cwnd, MIN_CWND)

<CODE ENDS>
```

CWND is updated differently depending on whether the congestion control is in fast increase state or not, as controlled by the variable in_fast_increase.

When in fast increase state, the congestion window is increased with the number of newly acknowledged bytes as long as the window is sufficiently used. Sparse feedback can potentially limit congestion window growth, an additional slack is therefore added, given by the number of newly acknowledged bytes.

The congestion window growth when in_fast_increase is false is dictated by the relation between qdelay and qdelay_target, congestion window growth is limited if the window is not used sufficiently.

SCReAM calculates the GAIN in a similar way to what is specified in [RFC6817]. However, [RFC6817] specifies that the CWND increase is limited by an additional function controlled by a constant ALLOWED_INCREASE. This additional limitation is removed in this specification.

Further the CWND is limited by max_bytes_in_flight and MIN_CWND. The limitation of the congestion window by the maximum number of bytes in flight over the last 5 seconds (max_bytes_in_flight) avoids possible over-estimation of the throughput after for example, idle periods. An additional MAX_BYTES_IN_FLIGHT_HEAD_ROOM allows for a slack, to allow for a certain amount of media coder output rate variability.

4.1.2.3. Competing flows compensation

It is likely that a flow using SCReAM algorithm will have to share congested bottlenecks with other flows that use a more aggressive congestion control algorithm, examples are large FTP flows using loss based congestion control. The worst condition occurs when the bottleneck queues are of tail-drop type with a large buffer size. SCReAM takes care of such situations by adjusting the qdelay_target when loss based flows are detected, as given by the pseudo code below.

```
<CODE BEGINS>
adjust_qdelay_target(qdelay)
  qdelay_norm_t = qdelay / QDELAY_TARGET_LOW
  update_qdelay_norm_history(qdelay_norm_t)
  # Compute variance
  qdelay_norm_var_t = VARIANCE(qdelay_norm_history(200))
  # Compensation for competing traffic
  # Compute average
  qdelay_norm_avg_t = AVERAGE(qdelay_norm_history(50))
  # Compute upper limit to target delay
  new_target_t = qdelay_norm_avg_t + sqrt(qdelay_norm_var_t)
  new_target_t *= QDELAY_TARGET_LO
  if (loss_event_rate > 0.002)
    # Packet losses detected
    qdelay_target = 1.5*new_target_t
  else
    if (qdelay_norm_var_t < 0.2)
      # Reasonably safe to set target qdelay
      qdelay_target = new_target_t
    else
      # Check if target delay can be reduced, this helps to avoid
      #  that the target delay is locked to high values for ever
      if (new_target_t < QDELAY_TARGET_LO)
        # Decrease target delay quickly as measured queueing
        #  delay is lower than target
        qdelay_target = max(qdelay_target*0.5,new_target_t)
      else
        # Decrease target delay slowly
        qdelay_target *= 0.9
      end
    end
  end

  # Apply limits
  qdelay_target = min(QDELAY_TARGET_HI, qdelay_target)
  qdelay_target = max(QDELAY_TARGET_LO, qdelay_target)
<CODE ENDS>
```

Two temporary variables are calculated. qdelay_norm_avg_t is the long term average queue delay, qdelay_norm_var_t is the long term variance of the queue delay. A high qdelay_norm_var_t indicates that the queue delay changes, this can be an indication of reduced bottleneck bandwidth or that a competing flow has just entered. Thus, it indicates that it is not safe to adjust the queue delay target.

A low qdelay_norm_var_t indicates that the queue delay is relatively stable, the reason can be that the queue delay is low but it can also be an indication that a competing flow is filling up the bottleneck

to the limit where packet losses may start to occur, in which case the queue delay will stay relatively high for a longer time.

The queue delay target is allowed to be increased if, either the loss event rate is above a given threshold or that qdelay_norm_var_t is low. Both these conditions indicate that a competing flow may be present. In all other cases the queue delay target is decreased.

The function that adjusts the qdelay_target is simple and has a certain risk to produce both false positive and negatives, The case that self-inflicted congestion by the SCReAM algorithm may be falsely interpreted as the presence of competing loss based FTP flows is a false positive. The opposite case where the algorithm fails to detect the presence of a competing FTP flow is a false negative.

Extensive simulations have shown that the algorithm performs well in LTE test cases and that it also performs well in simple bandwidth limited bottleneck test cases with competing FTP flows. It can however not be completely ruled out that this algorithm can fail. Especially the false positives can be problematic as the end to end delay can increase dramatically if the target queue delay is increased by accident as a result of self-inflicted congestion.

If it is deemed unlikely that competing flows occur over the same bottleneck, the algorithm described in this section MAY be turned off. One such case can be QoS enabled bearers in 3GPP based access such as LTE. However, when sending over the Internet, often the network conditions are not known for sure and it is in general not possible to make safe assumptions on how a network is used and whether or not competing flows share the same bottleneck. Therefore turning this algorithm off must be considered with caution as that can lead to basically zero throughput if competing with other, loss based, traffic.

4.1.2.4. Lost packet detection

Lost packet detection is based on the received sequence number list. A reordering window SHOULD be applied to avoid that packet reordering triggers loss events.
The reordering window is specified as a time unit, similar to the ideas behind RACK (Recent ACKnowledgement) [I-D.ietf-tcpm-rack]. The computation of the reordering window is made possible by means of a lost flag in the list of transmitted RTP packets. This flag is set if the received sequence number list indicates that the given RTP packet is missing. If a later feedback indicates that a previously lost marked packet was indeed received, then the reordering window is updated to reflect the reordering delay. The reordering window is given by the difference in time between the event that the packet was

marked as lost and the event that it was indicated as successfully received.
Loss is detected if a given RTP packet is not acknowledged within a time window (indicated by the reordering window) after an RTP packet with higher sequence number was acknowledged.

4.1.2.5. Send window calculation

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

- Bitrate variations: Media sources such as video encoders generally produce frames whose size always vary to a larger or smaller extent. The RTP queue absorbs the natural variations in frame sizes. The RTP queue should however be as short as possible, to avoid that the end to end delay increases. To achieve that, the media rate control takes the RTP queue size into account when the target bitrate for the media is computed. A strict 'send only when bytes in flight is less than the congestion window' rule can lead to that the RTP queue grows simply because the send window is limited, the effect of which would be that the target bitrate is pushed down. The consequence of this is that the congestion window will not increase, or will increase very slowly, because the congestion window is only allowed to increase when there is a sufficient amount of data in flight. The end effect is then that the media bitrate increases very slowly or not at all.

- Reverse (feedback) path congestion: Especially in transport over buffer-bloated networks, the one way delay in the reverse direction can jump due to congestion. The effect of this is that the acknowledgements are delayed with the result that the self-clocking is temporarily halted, even though the forward path is not congested.

The send window is adjusted depending on qdelay and its relation to the qdelay target and the relation between the congestion window and the number of bytes in flight. A strict rule is applied when qdelay is higher than qdelay_target, to avoid further queue buildup in the network. For cases when qdelay is lower than the qdelay_target, a more relaxed rule is applied. This allows the bitrate to increase quickly when no congestion is detected while still being able to give a stable behavior in congested situations.

The send window is given by the relation between the adjusted congestion window and the amount of bytes in flight according to the pseudo code below.

```
<CODE BEGINS>
calculate_send_window(qdelay, qdelay_target)
  # send window is computed differently depending on congestion level
  if (qdelay <= qdelay_target)
    send_wnd = cwnd+MSS-bytes_in_flight
  else
    send_wnd = cwnd-bytes_in_flight
  end
<CODE ENDS>
```

The send window is updated whenever an RTP packet is transmitted or an RTCP feedback messaged is received.

4.1.2.6. Packet pacing

Packet pacing is used in order to mitigate coalescing i.e. that packets are transmitted in bursts, with the increased risk of more jitter and potentially increased packet loss. Packet pacing also mitigates possible issues with queue overflow due to key-frame generation in video coders. The time interval between consecutive packet transmissions is enforced to be equal to or higher than t_pace where t_pace is given by the equations below :

```
<CODE BEGINS>
pace_bitrate = max (RATE_PACE_MIN, cwnd* 8 / s_rtt)
t_pace = rtp_size * 8 / pace_bitrate
<CODE ENDS>
```

rtp_size is the size of the last transmitted RTP packet, s_rtt is the smoothed round trip time. RATE_PACE_MIN is the minimum pacing rate.

4.1.2.7. Resuming fast increase

Fast increase can resume in order to speed up the bitrate increase in case congestion abates. The condition to resume fast increase (in_fast_increase = true) is that qdelay_trend is less than QDELAY_TREND_LO for T_RESUME_FAST_INCREASE seconds or more.

4.1.2.8. Stream prioritization

The SCReAM algorithm makes a good distinction between network congestion control and the media rate control. This is easily extended to many streams, in which case RTP packets from two or more RTP queues are scheduled at the rate permitted by the network congestion control.

The scheduling can be done by means of a few different scheduling regimes. For example the method applied in

[I-D.ietf-rmcat-coupled-cc] can be used. The implementation of SCReAM [SCReAM-CPP-implementation] use credit based scheduling. In credit based scheduling, credit is accumulated by queues as they wait for service and are spent while the queues are being serviced. For instance, if one queue is allowed to transmit 1000bytes, then a credit of 1000bytes is allocated to the other unscheduled queues. This principle can be extended to weighted scheduling in which case the credit allocated to unscheduled queues depends on the relative weights. The latter is also implemented in [SCReAM-CPP-implementation].

4.1.3. Media rate control

The media rate control algorithm is executed at regular intervals RATE_ADJUSTMENT_INTERVAL, with the exception of a prompt reaction to loss events. The media rate control operates based on the size of the RTP packet send queue and observed loss events. In addition, qdelay_trend is also considered in the media rate control to reduce the amount of induced network jitter.

The role of the media rate control is to strike a reasonable balance between a low amount of queuing in the RTP queue(s) and a sufficient amount of data to send in order to keep the data path busy. A too cautious setting leads to possible under-utilization of network capacity leading to that the flow can become starved out by other more opportunistic traffic. On the other hand, a too aggressive setting leads to increased jitter.

The target_bitrate is adjusted depending on the congestion state. The target bitrate can vary between a minimum value (TARGET_BITRATE_MIN) and a maximum value (TARGET_BITRATE_MAX). TARGET_BITRATE_MIN SHOULD be chosen to a low enough value to avoid that RTP packets become queued up when the network throughput is reduced. The sender SHOULD also be equipped with a mechanism that discards RTP packets in cases where the network throughput becomes very low and RTP packets are excessively delayed.

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

- rate_transmit: The measured transmit bitrate.
- rate_ack: The ACKed bitrate, i.e. the volume of ACKed bits per second.

Both estimates are updated every 200ms.

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

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

```
<CODE BEGINS>
# The target_bitrate is updated at a regular interval according
# to RATE_ADJUST_INTERVAL

on loss:
   # Loss event detected
   target_bitrate = max(BETA_R* target_bitrate, TARGET_BITRATE_MIN)
   exit
on ecn_mark:
   # ECN event detected
   target_bitrate = max(BETA_ECN* target_bitrate, TARGET_BITRATE_MIN)
   exit

ramp_up_speed_t = min(RAMP_UP_SPEED, target_bitrate/2.0)
scale_t = (target_bitrate - target_bitrate_last_max)/
     target_bitrate_last_max
scale_t = max(0.2, min(1.0, (scale_t*4)^2))
# min scale_t value 0.2 as the bitrate should be allowed to
#  increase at least slowly --> avoid locking the rate to
#  target_bitrate_last_max
if (in_fast_increase = true)
   increment_t = ramp_up_speed_t*RATE_ADJUST_INTERVAL
   increment_t *= scale_t
   target_bitrate += increment_t
else
   current_rate_t = max(rate_transmit, rate_ack)
   # Compute a bitrate change
   delta_rate_t = current_rate_t*(1.0-PRE_CONGESTION_GUARD*
        queue_delay_trend)-TX_QUEUE_SIZE_FACTOR *rtp_queue_size
   # Limit a positive increase if close to target_bitrate_last_max
   if (delta_rate_t > 0)
     delta_rate_t *= scale_t
     delta_rate_t =
       min(delta_rate_t,ramp_up_speed_t*RATE_ADJUST_INTERVAL)
   end
   target_bitrate += delta_rate_t
```

```
      # Force a slight reduction in bitrate if RTP queue
      #  builds up
      rtp_queue_delay_t = rtp_queue_size/current_rate_t
      if (rtp_queue_delay_t > RTP_QDELAY_TH)
        target_bitrate *= TARGET_RATE_SCALE_RTP_QDELAY
      end
   end

   rate_media_limit_t =
      max(current_rate_t, max(rate_media,rtp_rate_median))
   rate_media_limit_t *= (2.0-qdelay_trend_mem)
   target_bitrate = min(target_bitrate, rate_media_limit_t)
   target_bitrate = min(TARGET_BITRATE_MAX,
      max(TARGET_BITRATE_MIN,target_bitrate))
   <CODE ENDS>
```

In case of a loss event the target_bitrate is updated and the rate change procedure is exited. Otherwise the rate change procedure continues. The rationale behind the rate reduction due to loss is that a congestion window reduction will take effect, a rate reduction pro actively avoids RTP packets being queued up when the transmit rate decreases due to the reduced congestion window. A similar rate reduction happens when ECN events are detected.

The rate update frequency is limited by RATE_ADJUST_INTERVAL, unless a loss event occurs. The value is based on experimentation with real life limitations in video coders taken into account [SCReAM-CPP-implementation]. A too short interval is shown to make the rate control loop in video coders more unstable, a too long interval makes the overall congestion control sluggish.

When in fast increase state (in_fast_increase=true), the bitrate increase is given by the desired ramp-up speed (RAMP_UP_SPEED) . The ramp-up speed is limited when the target bitrate is low to avoid rate oscillation at low bottleneck bitrates. The setting of RAMP_UP_SPEED depends on preferences, a high setting such as 1000kbps/s makes it possible to quickly get high quality media, this is however at the expense of a increased jitter, which can manifest itself as e.g. choppy video rendering.

When in_fast_increase is false, the bitrate increase is given by the current bitrate and is also controlled by the estimated RTP queue and the qdelay trend, thus it is sufficient that an increased congestion level is sensed by the network congestion control to limit the bitrate. The target_bitrate_last_max is updated when congestion is detected.

Finally the target_bitrate is enforced to be within the defined min and max values.

The aware reader may notice the dependency on the qdelay in the computation of the target bitrate, this manifests itself in the use of the qdelay_trend. As these parameters are used also in the network congestion control one may suspect some odd interaction between the media rate control and the network congestion control, this is in fact the case if the parameter PRE_CONGESTION_GUARD is set to a high value. The use of qdelay_trend in the media rate control is solely to reduce jitter, the dependency can be removed by setting PRE_CONGESTION_GUARD=0, the effect is a somewhat faster rate increase after congestion, at the expense of increased jitter in congested situations.

4.2. SCReAM Receiver

The simple task of the SCReAM receiver is to feedback acknowledgements of received packets and total ECN count to the SCReAM sender, in addition, the receive time of the RTP packet with the highest sequence number is echoed back. Upon reception of each RTP packet the receiver MUST maintain enough information to send the aforementioned values to the SCReAM sender via a RTCP transport layer feedback message. The frequency of the feedback message depends on the available RTCP bandwidth. The requirements on the feedback elements and the feedback interval is described.

4.2.1. Requirements on feedback elements

The following feedback elements are REQUIRED for the basic functionality in SCReAM.

- A list of received RTP packets. This list SHOULD be sufficiently long to cover all received RTP packets. This list can be realized with the Loss RLE report block in [RFC3611].

- A wall clock timestamp corresponding to the received RTP packet with the highest sequence number is required in order to compute the qdelay. This can be realized by means of the Packet Receipt Times Report Block in [RFC3611]. begin_seq MUST be set to the highest received (possibly wrapped around) sequence number, end_seq MUST be set to begin_seq+1 % 65536. The timestamp clock MAY be set according to [RFC3611] i.e. equal to the RTP timestamp clock. Detailed individual packet receive times is not necessary as SCReAM does currently not describe how this can be used.

The basic feedback needed for SCReAM involves the use of the Loss RLE report block and the Packet Receipt Times block defined in Figure 2.

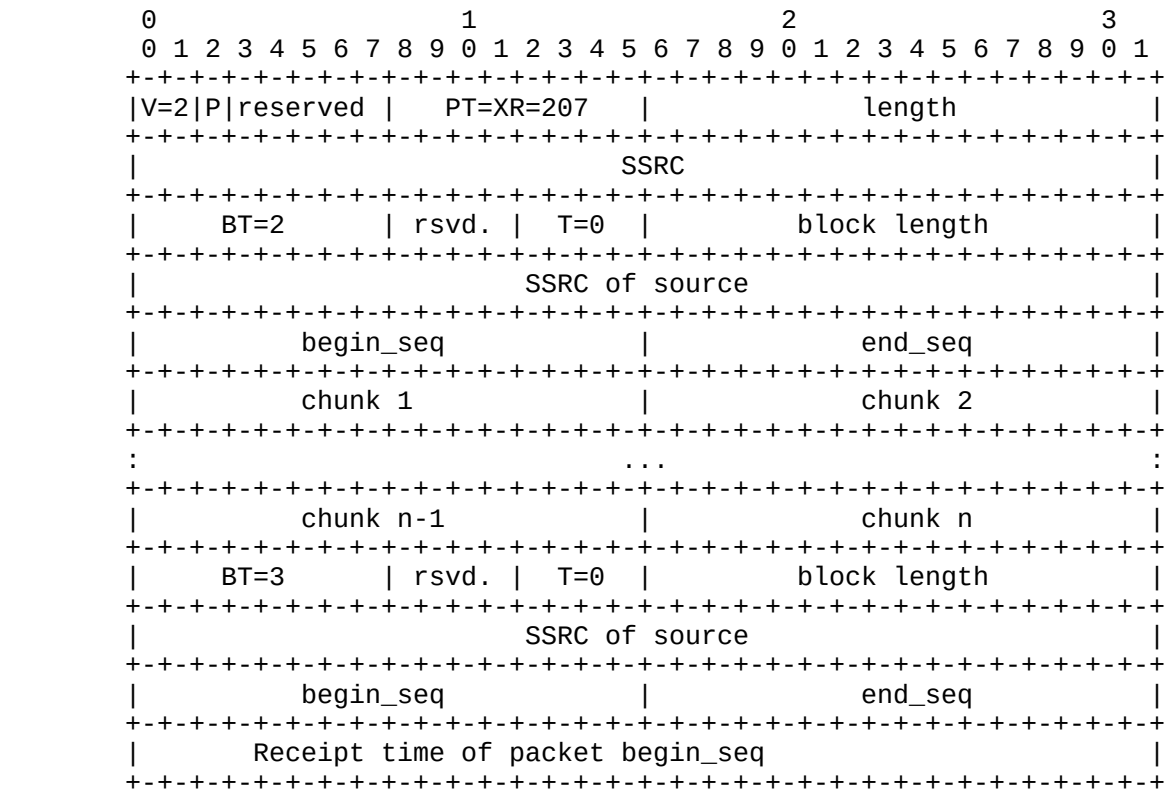


Figure 2: Basic feedback message for SCReAM, based on RFC3611

In a typical use case, no more than four Loss RLE chunks are needed, thus the feedback message will be 44bytes. It is obvious from the figure that there is a lot of redundant information in the feedback message. A more optimized feedback format, including the additional feedback elements listed below, could reduce the feedback message size a bit.

Additional feedback elements that can improve the performance of SCReAM are:

- Accumulated number of ECN-CE marked packets (n_ECN). This can for instance be realized with the ECN Feedback Report Format in [RFC6679]. The given feedback report format is actually a slight overkill as SCReAM would do quite well with only a counter that increments by one for each received packet with the ECN-CE code point set. The more bulky format could nevertheless be useful for e.g ECN black-hole detection.

4.2.2. Requirements on feedback intensity

SCReAM benefits from a relatively frequent feedback. It is RECOMMENDED that a SCReAM implementation follows the guidelines below.

The feedback interval depends on the media bitrate. At low bitrates it is sufficient with a feedback interval of 100 to 400ms, while at high bitrates a feedback interval of roughly 20ms is to prefer, at very high bitrates, even shorter feedback intervals MAY be needed in order to keep the self-clocking in SCReAM working well. One piece of evidence of a too sparse feedback is that the SCReAM implementation cannot reach high bitrates, even in uncongested links. A more frequent feedback might solve this issue.

The numbers above can be formulated as feedback interval function that can be useful for the computation of the desired RTCP bandwidth. The following equation expresses the feedback rate:

```
rate_fb = min(50,max(2.5,rate_media/10000))
```

rate_media is the RTP media bitrate expressed in [bits/s], rate_fb is the feedback rate expressed in [packets/s]. Converted to feedback interval we get:

```
fb_int = 1.0/min(50,max(2.5,rate_media/10000))
```

The transmission interval is not critical, this means that in the case of multi-stream handling between two hosts, the feedback for two or more SSRCs can be bundled to save UDP/IP overhead, the final realized feedback interval SHOULD however not exceed 2*fb_int in such cases meaning that a scheduled feedback transmission event should not be delayed more that fb_int.

SCReAM works with AVPF regular mode, immediate or early mode is not required by SCReAM but can nonetheless be useful for e.g RTCP messages not directly related to SCReAM, such as those specified in [RFC4585]. It is RECOMMENDED to use reduced size RTCP [RFC5506] where regular full compound RTCP transmission is controlled by trr-int as described in [RFC4585].

5. Discussion

This section covers a few discussion points

- Clock drift: SCReAM can suffer from the same issues with clock drift as is the case with LEDBAT [RFC6817]. Section A.2 in [RFC6817] however describes ways to mitigate issues with clock drift.

- Support for alternate ECN semantics: This specification adopts the proposal in [I-D.ietf-tcpm-alternativebackoff-ecn] to reduce the congestion window less when ECN based congestion events are detected. Future work on Low Loss Low Latency for Scalable throughput (L4S) may lead to updates in a future RFC that describes SCReAM support for L4S.

- A new RFC4585 transport layer feedback message could to be standardized if the use of the already existing RTCP extensions as described in Section 4.2 is not deemed sufficient.

- The target bitrate given by SCReAM depicts the bitrate including RTP and FEC overhead. The media encoder SHOULD take this overhead into account when the media bitrate is set. This means that the media coder bitrate SHOULD be computed as

  media_rate = target_bitrate - rtp_plus_fec_overhead_bitrate

  It is not strictly necessary to make a 100% perfect compensation for the overhead as the SCReAM algorithm will inherently compensate for moderate errors. Under-compensation of the overhead has the effect of increasing jitter while overcompensation will have the effect of causing the bottleneck link to become under-utilized.

6. Implementation status

[Editor’s note: Please remove the whole section before publication, as well reference to RFC 7942]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and MUST NOT be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations MAY exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see it".

6.1. OpenWebRTC

The SCReAM algorithm has been implemented in the OpenWebRTC project [OpenWebRTC], an open source WebRTC implementation from Ericsson Research. This SCReAM implementation is usable with any WebRTC endpoint using OpenWebRTC.

- Organization : Ericsson Research, Ericsson.
- Name : OpenWebRTC gst plug-in.
- Implementation link : The GStreamer plug-in code for SCReAM can be found at github repository [SCReAM-implementation] The wiki (https://github.com/EricssonResearch/openwebrtc/wiki) contains required information for building and using OpenWebRTC.
- Coverage : The code implements the specification in this memo. The current implementation has been tuned and tested to adapt a video stream and does not adapt the audio streams.
- Implementation experience : The implementation of the algorithm in the OpenWebRTC has given great insight into the algorithm itself and its interaction with other involved modules such as encoder, RTP queue etc. In fact it proves the usability of a self-clocked rate adaptation algorithm in the real WebRTC system. The implementation experience has led to various algorithm improvements both in terms of stability and design. The current implementation use an n_loss counter for lost packets indication, this is subject to change in later versions to a list of received RTP packets.
- Contact : irc://chat.freenode.net/openwebrtc

6.2. A C++ Implementation of SCReAM

- Organization : Ericsson Research, Ericsson.
- Name : SCReAM.
- Implementation link : A C++ implementation of SCReAM is available at[SCReAM-CPP-implementation]. The code includes full support for

congestion control, rate control and multi stream handling, it can be integrated in web clients given the addition of extra code to implement the RTCP feedback and RTP queue(s). The code also includes a rudimentary implementation of a simulator that allows for some initial experiments. An additional experiment with SCReAM in a remote control arrangement is also documented.

- Coverage : The code implements the specification in this memo.

- Contact : ingemar.s.johansson@ericsson.com

7. Suggested experiments

SCReAM has been evaluated in a number of different ways, most of the evaluation has been in simulator. The OpenWebRTC implementation work involved extensive testing with artificial bottlenecks with varying bandwidths and using two different video coders (OpenH264 and VP9), the experience of this lead to further improvements of the media rate control logic.

Further experiments are preferably done by means of implementation in real clients and web browsers. RECOMMENDED experiments are:

- Trials with various access technologies: EDGE/3G/4G, WiFi, DSL. Some experiments have already been carried out with LTE access, see e.g. [SCReAM-CPP-implementation] and [SCReAM-implementation-experience]

- Trials with different kinds of media: Audio, Video, slide show content. Evaluation of multi stream handling in SCReAM.

- Evaluation of functionality of competing flows compensation mechanism: Evaluate how SCReAM performs with competing TCP like traffic and to what extent the competing flows compensation causes self-inflicted congestion.

- Determine proper parameters: A set of default parameters are given that makes SCReAM work over a reasonably large operation range, however for instance for very low or very high bitrates it may be necessary to use different values for instance for the RAMP_UP_SPEED.

- Experimentation with further improvements to the congestion window and media bitrate calculation. [SCReAM-CPP-implementation] implements some optimizations, not described in this memo, that improve performance slightly. Further experiments are likely to lead to more optimizations of the algorithm.

8. Acknowledgements

We would like to thank the following persons for their comments, questions and support during the work that led to this memo: Markus Andersson, Bo Burman, Tomas Frankkila, Frederic Gabin, Laurits Hamm, Hans Hannu, Nikolas Hermanns, Stefan Haakansson, Erlendur Karlsson, Daniel Lindstroem, Mats Nordberg, Jonathan Samuelsson, Rickard Sjoeberg, Robert Swain, Magnus Westerlund, Stefan Aalund. Many additional thanks to RMCAT chairs Karen E. E. Nielsen and Mirja Kuehlewind for patiently reading, suggesting improvements and also for asking all the difficult but necessary questions. Thanks to Stefan Holmer, Xiaoqing Zhu, Safiqul Islam and David Hayes for the additional review of this document. Thanks to Ralf Globisch for taking time to try out SCReAM in his challenging low bitrate use cases, Robert Hedman for finding a few additional flaws in the running code, and Gustavo Garcia and 'miseri' for code contributions.

9. IANA Considerations

There is currently no request to IANA

10. Security Considerations

The feedback can be vulnerable to attacks similar to those that can affect TCP. It is therefore RECOMMENDED that the RTCP feedback is at least integrity protected. Furthermore, as SCReAM is self-clocked, a malicious middlebox can drop RTCP feedback packets and thus cause the self-clocking in SCReAM to stall. This attack is however mitigated by the minimum send rate maintained by SCReAM when no feedback is received.

11. Change history

A list of changes:

- WG-12 to WG-13: IESG comments addressed
- WG-11 to WG-12: Review comments from Joel Halpern and Mirja
- WG-10 to WG-11: Review comments from Mirja
- WG-9 to WG-10: Minor edits
- WG-08 to WG-09: Updated based shepherd review by Martin Stiemerling, Q-bit semantics are removed as this is superfluous for the moment. Pacing and RTCP considerations are moved up from the appendix, FEC discussion moved to discussion section.

- WG-07 to WG-08: Avoid draft expiry
- WG-06 to WG-07: Updated based on WGLC review by David Hayes and Safiqul Islam
- WG-05 to WG-06: Added list of suggested experiments
- WG-04 to WG-05: Congestion control and rate control simplified somewhat
- WG-03 to WG-04: Editorial fixes
- WG-02 to WG-03: Review comments from Stefan Holmer and Xiaoqing Zhu addressed, owd changed to qdelay for clarity. Added appendix section with RTCP feedback requirements, including a suggested basic feedback format based Loss RLE report block and the Packet Receipt Times blocks in [RFC3611]. Loss detection added as a section. Transmission scheduling and packet pacing explained in appendix. Source quench semantics added to appendix.
- WG-01 to WG-02: Complete restructuring of the document. Moved feedback message to a separate draft.
- WG-00 to WG-01 : Changed the Source code section to Implementation status section.
- -05 to WG-00 : First version of WG doc, moved additional features section to Appendix. Added description of prioritization in SCReAM. Added description of additional cap on target bitrate
- -04 to -05 : ACK vector is replaced by a loss counter, PT is removed from feedback, references to source code added
- -03 to -04 : Extensive changes due to review comments, code somewhat modified, frame skipping made optional
- -02 to -03 : Added algorithm description with equations, removed pseudo code and simulation results
- -01 to -02 : Updated GCC simulation results
- -00 to -01 : Fixed a few bugs in example code

12. References

12.1. Normative References

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

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

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/rfc4585>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <https://www.rfc-editor.org/info/rfc5506>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP’s Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/info/rfc6298>.

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

12.2. Informative References

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

[I-D.ietf-rmcat-wireless-tests]
Sarker, Z., Johansson, I., Zhu, X., Fu, J., Tan, W., and M. Ramalho, "Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks", draft-ietf-rmcat-wireless-tests-04 (work in progress), May 2017.

[I-D.ietf-tcpm-alternativebackoff-ecn]
Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", draft-ietf-tcpm-alternativebackoff-ecn-02 (work in progress), October 2017.

[I-D.ietf-tcpm-rack]
Cheng, Y., Cardwell, N., and N. Dukkipati, "RACK: a time-based fast loss detection algorithm for TCP", draft-ietf-tcpm-rack-02 (work in progress), March 2017.

[LEDBAT-delay-impact]
"Assessing LEDBAT’s Delay Impact, IEEE communications letters, vol. 17, no. 5, May 2013", May 2013, <http://home.ifi.uio.no/michawe/research/publications/ledbat-impact-letters.pdf>.

[OpenWebRTC]
"Open WebRTC project.", <http://www.openwebrtc.io/>.

[Packet-conservation]
"Congestion Avoidance and Control, ACM SIGCOMM Computer Communication Review 1988", 1988.

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

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

[RFC7478] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", RFC 7478, DOI 10.17487/RFC7478, March 2015, <https://www.rfc-editor.org/info/rfc7478>.

[RFC7661] Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to Support Rate-Limited Traffic", RFC 7661, DOI 10.17487/RFC7661, October 2015, <https://www.rfc-editor.org/info/rfc7661>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[SCReAM-CPP-implementation] "C++ Implementation of SCReAM", <https://github.com/EricssonResearch/scream>.

[SCReAM-implementation] "SCReAM Implementation", <https://github.com/EricssonResearch/openwebrtc-gst-plugins>.

[SCReAM-implementation-experience] "Updates on SCReAM : An implementation experience", <https://www.ietf.org/proceedings/94/slides/slides-94-rmcat-8.pdf>.

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

Authors' Addresses

Ingemar Johansson
Ericsson AB
Laboratoriegraend 11
Luleaa 977 53
Sweden

Phone: +46 730783289
Email: ingemar.s.johansson@ericsson.com

Zaheduzzaman Sarker
Ericsson AB
Laboratoriegraend 11
Luleaa 977 53
Sweden

Phone: +46 761153743
Email: zaheduzzaman.sarker@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 23, 2019

X. Zhu
S. Mena
Cisco Systems
Z. Sarker
Ericsson AB

February 19, 2019

Video Traffic Models for RTP Congestion Control Evaluations
draft-ietf-rmcat-video-traffic-model-07

Abstract

This document describes two reference video traffic models for evaluating RTP congestion control algorithms. The first model statistically characterizes the behavior of a live video encoder in response to changing requests on the target video rate. The second model is trace-driven and emulates the output of actual encoded video frame sizes from a high-resolution test sequence. Both models are designed to strike a balance between simplicity, repeatability, and authenticity in modeling the interactions between a live video traffic source and the congestion control module. Finally, the document describes how both approaches can be combined into a hybrid model.

Table of Contents

1. Introduction

When evaluating candidate congestion control algorithms designed for real-time interactive media, it is important to account for the characteristics of traffic patterns generated from a live video encoder. Unlike synthetic traffic sources that can conform perfectly to the rate changing requests from the congestion control module, a live video encoder can be sluggish in reacting to such changes. The output rate of a live video encoder also typically deviates from the target rate due to uncertainties in the encoder rate control process.

Consequently, end-to-end delay and loss performance of a real-time media flow can be further impacted by rate variations introduced by the live encoder.

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

One way to strike a balance between the above considerations is to evaluate congestion control algorithms using a synthetic video traffic source model that captures key characteristics of the behavior of a live video encoder. The synthetic traffic model should also contain tunable parameters so that it can be flexibly adjusted to reflect the wide variations in real-world live video encoder behaviors. To this end, this draft presents two reference models. The first is based on statistical modeling. The second is driven by frame size and interval traces recorded from a real-world encoder. The draft also discusses the pros and cons of each approach, as well as how both approaches can be combined into a hybrid model.

2. Terminology

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

3. Desired Behavior of A Synthetic Video Traffic Model

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

Due to uncertainties in the captured video scene, the output rate typically deviates from the specified target. In the presence of a significant change in target rate, the encoder’s output frame sizes sometimes fluctuate for a short, transient period of time before the output rate converges to the new target. Finally, while most of the frames in a live session are encoded in predictive mode (i.e., P-frames in [H264]), the encoder can occasionally generate a large intra-coded frame (i.e., I-frame as defined in [H264]) or a frame

partially containing intra-coded blocks in an attempt to recover from losses, to re-sync with the receiver, or during the transient period of responding to target rate or spatial resolution changes.

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

- To change bitrate. This includes the ability to change framerate and/or spatial resolution or to skip frames upon request.
- To fluctuate around the target bitrate specified by the congestion control module.
- To show a delay in convergence to the target bitrate.
- To generate intra-coded or repair frames on demand.

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

- Low computational complexity: The model should be computationally lightweight, otherwise it defeats the whole purpose of serving as a substitute for a live video encoder.
- Temporal pattern similarity: The individual traffic trace instances generated by the model should mimic the temporal pattern of those from a real video encoder.
- Statistical resemblance: The synthetic traffic source should match the outcome of the real video encoder in terms of statistical characteristics, such as the mean, variance, peak, and autocorrelation coefficients of the bitrate. It is also important that the statistical resemblance should hold across different time scales, ranging from tens of milliseconds to sub-seconds.
- A wide range of coverage: The model should be easily configurable to cover a wide range of codec behaviors (e.g., with either fast or slow reaction time in live encoder rate control) and video content variations (e.g., ranging from high to low motion).

These distinct behavior features can be characterized via simple statistical modeling or a trace-driven approach. Section 5 and Section 6 provide an example of each approach, respectively. Section 7 discusses how both models can be combined together.

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

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

The synthetic video source dynamically generates a sequence of dummy video frames with varying size and interval. These dummy frames are processed by other modules in order to transmit the video stream over the network. During the lifetime of a video transmission session, the synthetic video source will typically be required to adapt its encoding bitrate, and sometimes the spatial resolution and frame rate.

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

- Target bitrate R_v: target bitrate request measured in bits per second (bps). Typically, the congestion control module calculates the target bitrate and updates it dynamically over time. Depending on the congestion control algorithm in use, the update requests can either be periodic (e.g., once per second), or on-demand (e.g., only when a drastic bandwidth change over the network is observed).

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

- Target frame resolution XY: the 2-dimensional vector indicating the preferred frame resolution in pixels. Several factors govern the resolution requested to the synthetic video source over time. Examples of such factors include the capturing resolution of the native camera and the display size of the destination screen. The target frame resolution also depends on the current target bitrate R_v, since it does not make sense to pair very low spatial resolutions with very high bitrates, and vice-versa.

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

- On-demand generation of intra (I) frame: the request to encode another I frame to avoid further error propagation at the receiver when severe packet losses are observed. This request typically comes from the error control module. It can be initiated either by the sender or by the receiver via Full Intra Request (FIR) messages as defined in [RFC5104].

An example of outgoing interface call --- marked as (b) in Figure 1 --- is the rate range [R_min, R_max]. Here, R_min and R_max are meant to capture the dynamic rate range and actual live video encoder is capable of generating given the input video content. This typically depends on the video content complexity and/or display type (e.g., higher R_max for video contents with higher motion complexity, or for displays of higher resolution). Therefore, these values will not change with R_v but may change over time if the content is changing.

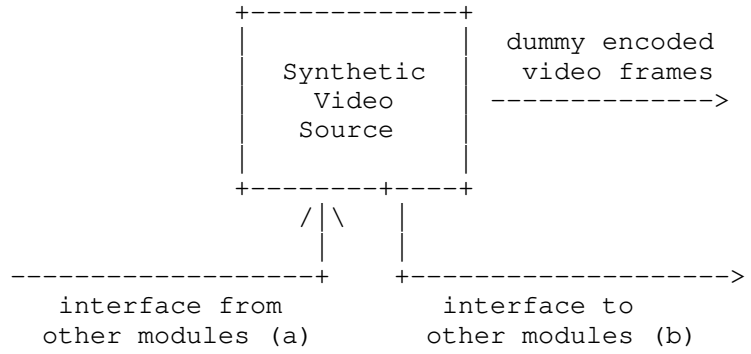


Figure 1: Interaction between synthetic video encoder and other modules at the sender

5. A Statistical Reference Model

This section describes one simple statistical model of the live video encoder traffic source. Figure 2 summarizes the list of tunable parameters in this statistical model. A more comprehensive survey of popular methods for modeling video traffic source behavior can be found in [Tanwir2013].

Notation	Parameter Name	Example Value
R_v	Target bitrate request	1 Mbps
FPS	Target frame rate	30 Hz
tau_v	Encoder reaction latency	0.2 s
K_d	Burst duration of the transient period	8 frames
K_B	Burst frame size during the transient period	13.5 KBytes*
t0	Reference frame interval 1/FPS	33 ms
B0	Reference frame size R_v/8/FPS	4.17 KBytes
SCALE_t	Scaling parameter of the zero-mean Laplacian distribution describing deviations in normalized frame interval (t-t0)/t0	0.15
SCALE_B	Scaling parameter of the zero-mean Laplacian distribution describing deviations in normalized frame size (B-B0)/B0	0.15
R_min	minimum rate supported by video encoder type or content activity	150 Kbps
R_max	maximum rate supported by video encoder type or content activity	1.5 Mbps

* Example value of K_B for a video stream encoded at 720p and 30 frames per second, using H.264/AVC encoder.

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

5.1. Time-damped response to target rate update

While the congestion control module can update its target bitrate request R_v at any time, the statistical model dictates that the encoder will only react to such changes tau_v seconds after a

previous rate transition. In other words, when the encoder has reacted to a rate change request at time t, it will simply ignore all subsequent rate change requests until time t+tau_v.

5.2. Temporary burst and oscillation during the transient period

The output bitrate R_o during the period [t, t+tau_v] is considered to be in a transient state when reacting to abrupt changes in target rate. Based on observations from video encoder output data, the encoder reaction to a new target bitrate request can be characterized by high variations in output frame sizes. It is assumed in the model that the overall average output bitrate R_o during this transient period matches the target bitrate R_v. Consequently, the occasional burst of large frames is followed by smaller-than-average encoded frames.

This temporary burst is characterized by two parameters:

- burst duration K_d: number of frames in the burst event; and
- burst frame size K_B: size of the initial burst frame which is typically significantly larger than average frame size at steady state.

It can be noted that these burst parameters can also be used to mimic the insertion of a large on-demand I frame in the presence of severe packet losses. The values of K_d and K_B typically depend on the type of video codec, spatial and temporal resolution of the encoded stream, as well as the video content activity level.

5.3. Output rate fluctuation at steady state

The output bitrate R_o during steady state is modeled as randomly fluctuating around the target bitrate R_v. The output traffic can be characterized as the combination of two random processes denoting the frame interval t and output frame size B over time, as the two major sources of variations in the encoder output. For simplicity, the deviations of t and B from their respective reference levels are modeled as independent and identically distributed (i.i.d) random variables following the Laplacian distribution [Papoulis]. More specifically:

- Fluctuations in frame interval: the intervals between adjacent frames have been observed to fluctuate around the reference interval of t0 = 1/FPS. Deviations in normalized frame interval DELTA_t = (t-t0)/t0 can be modeled by a zero-mean Laplacian distribution with scaling parameter SCALE_t. The value of SCALE_t dictates the "width" of the Laplacian distribution and therefore

the amount of fluctuation in actual frame intervals (t) with respect to the reference frame interval t0.

- Fluctuations in frame size: the output encoded frame sizes also tend to fluctuate around the reference frame size B0=R_v/8/FPS. Likewise, deviations in the normalized frame size DELTA_B = (B-B0)/B0 can be modeled by a zero-mean Laplacian distribution with scaling parameter SCALE_B. The value of SCALE_B dictates the "width" of this second Laplacian distribution and correspondingly the amount of fluctuations in output frame sizes (B) with respect to the reference target B0.

Both values of SCALE_t and SCALE_B can be obtained via parameter fitting from empirical data captured for a given video encoder. Example values are listed in Figure 2 based on empirical data presented in [IETF-Interim].

5.4. Rate range limit imposed by video content

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

6. A Trace-Driven Model

The second approach for modeling a video traffic source is trace-driven. This can be achieved by running an actual live video encoder on a set of chosen raw video sequences and using the encoder's output traces for constructing a synthetic video source. With this approach, the recorded video traces naturally exhibit temporal fluctuations around a given target bitrate request R_v from the congestion control module.

The following list summarizes the main steps of this approach:

1. Choose one or more representative raw video sequences.

2. Encode the sequence(s) using an actual live video encoder. Repeat the process for a number of bitrates. Keep only the sequence of frame sizes for each bitrate.

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

4. Upon a target bitrate request R_v from the controller, look up the closest bitrates among those previously stored. Use the

frame size sequences stored for those bitrates to approximate the frame sizes to output.

5. The output of the synthetic video traffic source contains "encoded" frames with dummy contents but with realistic sizes.

In the following, Section 6.1 explains the first three steps (1-3), Section 6.2 elaborates on the remaining two steps (4-5). Finally, Section 6.3 briefly discusses the possibility to extend the trace-driven model for supporting time-varying frame rate and/or time-varying frame resolution.

6.1. Choosing the video sequence and generating the traces

The first step is a careful choice of a set of video sequences that are representative of the target use cases for the video traffic model. For the example use case of interactive video conferencing, it is recommended to choose a sequence with content that resembles a "talking head", e.g. from a news broadcast or recording of an actual video conferencing call.

The length of the chosen video sequence is a tradeoff. If it is too long, it will be difficult to manage the data structures containing the traces. If it is too short, there will be an obvious periodic pattern in the output frame sizes, leading to biased results when evaluating congestion control performance. It has been empirically determined that a sequence 2 to 4 minutes in length sufficiently avoids the periodic pattern.

Given the chosen raw video sequence, denoted S, one can use a live encoder, e.g. some implementation of [H264] or [HEVC], to produce a set of encoded sequences. As discussed in Section 3, the output bitrate of the live encoder can be achieved by tuning three input parameters: quantization step size, frame rate, and picture resolution. In order to simplify the choice of these parameters for a given target rate, one can typically assume a fixed frame rate (e.g. 30 fps) and a fixed resolution (e.g., 720p) when configuring the live encoder. See Section 6.3 for a discussion on how to relax these assumptions.

Following these simplifications, the chosen encoder can be configured to start at a constant target bitrate, then vary the quantization step size (internally via the video encoder rate controller) to meet various externally specified target rates. It can be further assumed the first frame is encoded as an I-frame and the rest are P-frames (see, e.g., [H264] for definitions of I- and P-frames). For live encoding, the encoder rate control algorithm typically does not use knowledge of frames in the future when encoding a given frame.

Given the minimum and maximum bitrates at which the synthetic codec is to operate (denoted as R_min and R_max, see Section 4), the entire range of target bitrates can be divided into n_s steps. This leads to a encoding bitrate ladder of (n_s + 1) choices equally spaced apart by the step length l = (R_max - R_min)/n_s. The following simple algorithm is used to encode the raw video sequence.

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

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

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

Finally, note that, as mentioned in previous sections, R_min and R_max may be modified after the initial sequences are encoded. Henceforth, for notational clarity, we refer to the bitrate range of the trace file as [Rf_min, Rf_max]. The algorithm described in the next section also covers the cases when the current target bitrate is less than Rf_min, or greater than Rf_max.

6.2. Using the traces in the synthetic codec

The main idea behind the trace-driven synthetic codec is that it mimics the rate adaptation behavior of a real live codec upon dynamic updates of the target bitrate request R_v by the congestion control module. It does so by switching to a different frame size vector stored in the map Traces when needed.

6.2.1. Main algorithm

The main algorithm for rate adaptation in the synthetic codec maintains two variables: r_current and t_current.

- The variable r_current points to one of the keys of map Traces. Upon a change in the value of R_v, typically because the congestion controller detects that the network conditions have changed, r_current is updated based on R_v as follows:

```
R_ref = min (Rf_max, max(Rf_min, R_v))

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

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

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

where operator % denotes modulo, and SkipFrames is a predefined constant that denotes the number of frames to be skipped at the beginning of frame size vectors after t_current has wrapped around. The point of constant SkipFrames is avoiding the effect of periodically sending a large I-frame followed by several smaller-than-average P-frames. A typical value of SkipFrames is 20, although it could be set to 0 if one is interested in studying the effect of sending I-frames periodically.

The initial value of r_current is set to R_min, and the initial value of t_current is set to 0.

When a new frame is due, its size can be calculated following one of the three cases below:

a) Rf_min <= R_v < Rf_max: the output frame size is calculated via linear interpolation of the frame sizes appearing in Traces[r_current] and Traces[r_current + 1]. The interpolation is done as follows:

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

b) R_v < Rf_min: the output frame size is calculated via scaling with respect to the lowest bitrate Rf_min in the trace file, as follows:

```
w = R_v / Rf_min
framesize = max(fs_min, factor * Traces[Rf_min][t_current])
```

c) R_v >= Rf_max: the output frame size is calculated by scaling with respect to the highest bitrate Rf_max in the trace file, as follows:

```
w = R_v / Rf_max
framesize = min(fs_max, w * Traces[Rf_max][t_current])
```

In cases b) and c), floating-point arithmetic is used for computing the scaling factor w. The resulting value of the instantaneous frame size (framesize) is further clipped within a reasonable range between fs_min (e.g., 10 bytes) and fs_max (e.g., 1MB).

6.2.2. Notes to the main algorithm

Note that the main algorithm as described above can be further extended to mimic some additional typical behaviors of a live video encoder. Two examples are given below:

- I-frames on demand: The synthetic codec can be extended to simulate the sending of I-frames on demand, e.g., as a reaction to losses. To implement this extension, the codec's incoming interface (see (a) in Figure 1) is augmented with a new function to request a new I-frame. Upon calling such function, t_current is reset to 0.

- Variable step length l between R_min and R_max: In the main algorithm, the step length l is fixed for ease of explanation. However, if the range [R_min, R_max] is very wide, it is also possible to define a set of intermediate encoding rates with variable step length. The rationale behind this modification is that the difference between 400 kbps and 600 kbps as target bitrate is much more significant than the difference between 4400 kbps and 4600 kbps. For example, one could define steps of length 200 Kbps under 1 Mbps, then steps of length 300 Kbps between 1 Mbps and 2 Mbps; 400 Kbps between 2 Mbps and 3 Mbps, and so on.

6.3. Varying frame rate and resolution

The trace-driven synthetic codec model explained in this section is relatively simple due to the choice of fixed frame rate and frame resolution. The model can be extended further to accommodate variable frame rate and/or variable spatial resolution.

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

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

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

7. Combining The Two Models

It is worthwhile noting that the statistical and trace-driven models each have their own advantages and drawbacks. Both models are fairly simple to implement. It takes significantly greater effort to fit the parameters of a statistical model to actual encoder output data. In contrast, it is straightforward for a trace-driven model to obtain encoded frame size data. Once validated, the statistical model is more flexible in mimicking a wide range of encoder/content behaviors by simply varying the corresponding parameters in the model. In this regard, a trace-driven model relies -- by definition -- on additional data collection efforts for accommodating new codecs or video contents.

In general, the trace-driven model is more realistic for mimicking the ongoing, steady-state behavior of a video traffic source with fluctuations around a constant target rate. In contrast, the statistical model is more versatile for simulating the behavior of a video stream in transient, such as when encountering sudden rate changes. It is also possible to combine both methods into a hybrid model. In this case, the steady-state behavior is driven by traces during steady state and the transient-state behavior is driven by the statistical model.

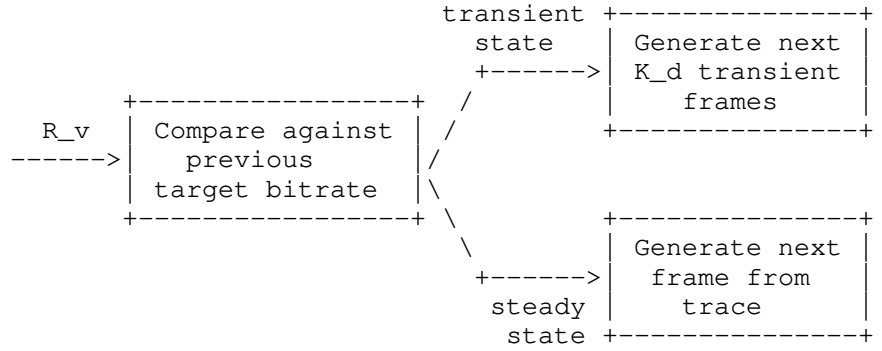


Figure 3: A hybrid video traffic model

As shown in Figure 3, the video traffic model operates in a transient state if the requested target rate R_v is substantially different from the previous target, or else it operates in steady state. During the transient state, a total of K_d frames are generated by the statistical model, resulting in one (1) big burst frame with size K_B followed by K_d-1 smaller frames. When operating at steady state, the video traffic model simply generates a frame according to the trace-driven model given the target rate, while modulating the frame interval according to the distribution specified by the statistical model. One example criterion for determining whether the traffic model should operate in a transient state is whether the rate change exceeds 10% of the previous target rate. Finally, as this model follows transient-state behavior dictated by the statistical model, upon a substantial rate change, the model will follow the time-damping mechanism as defined in Section 5.1, which is governed by parameter tau_v.

8. Implementation Status

The statistical, trace-driven, and hybrid models as described in this draft have been implemented as a stand-alone, platform-independent synthetic traffic source module. It can be easily integrated into network simulation platforms such as [ns-2] and [ns-3], as well as testbeds using a real network. The stand-alone traffic source module is available as an open source implementation at [Syncodecs].

9. IANA Considerations

There are no IANA impacts in this memo.

10. Security Considerations

The synthetic video traffic models as described in this draft do not impose any security threats. They are designed to mimic realistic traffic patterns for evaluating candidate RTP-based congestion control algorithms, so as to ensure stable operations of the network. It is RECOMMENDED that candidate algorithms be tested using the video traffic models presented in this draft before wide deployment over the Internet. If the generated synthetic traffic flows are sent over the Internet, they also need to be congestion controlled.

11. References

11.1. Normative References

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

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

11.2. Informative References

[H264] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services", May 2003, <https://www.itu.int/rec/T-REC-H.264>.

[HEVC] ITU-T Recommendation H.265, "High efficiency video coding", April 2013, <https://www.itu.int/rec/T-REC-H.265>.

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

[IETF-Interim] Zhu, X., Mena, S., and Z. Sarker, "Update on RMCAT Video Traffic Model: Trace Analysis and Model Update", April 2017, <https://www.ietf.org/proceedings/interim-2017-rmcat-01/slides/slides-interim-2017-rmcat-01-sessa-update-on-video-traffic-model-draft-00.pdf>.

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

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

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

[Papoulis] Papoulis, A., "Probability, Random Variables and Stochastic Processes", 2002.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <https://www.rfc-editor.org/info/rfc5104>.

[Syncodecs] Mena, S., D'Aronco, S., and X. Zhu, "Syncodecs: Synthetic codecs for evaluation of RMCAT work", <https://github.com/cisco/syncodecs>.

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

Authors' Addresses

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

Email: xiaoqzhu@cisco.com

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

Email: semena@cisco.com

Zaheduzzaman Sarker
Ericsson AB
Luleae, SE 977 53
Sweden

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 8, 2016

Z. Sarker
I. Johansson
Ericsson AB
X. Zhu
J. Fu
W. Tan
M. Ramalho
Cisco Systems
May 7, 2016

Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks
draft-ietf-rmcat-wireless-tests-02

Abstract

It is evident that to ensure seamless and robust user experience across all type of access networks multimedia communication suits should adapt to the changing network conditions. There is an ongoing effort in IETF RMCAT working group to standardize rate adaptive algorithm(s) to be used in the real-time interactive communication. In this document test cases are described to evaluate the performances of the proposed endpoint adaptation solutions in LTE networks and Wi-Fi networks. The proposed algorithms should be evaluated using the test cases defined in this document to select most optimal solutions.

Status of This Memo

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

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

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

This Internet-Draft will expire on November 8, 2016.

Table of Contents

1. Introduction

Wireless networks (both cellular and Wi-Fi [IEEE802.11] local area network) are an integral part of the Internet. Mobile devices connected to the wireless networks produces huge amount of media traffic in the Internet. They covers the scenarios of having a video call in the bus to media consumption sitting on a couch in a living room. It is a well known fact that the characteristic and challenges for offering service over wireless network are very different than providing the same over a wired network. Even though RMCAT basic test cases defines number of test cases that covers lots of effects of the impairments visible in the wireless networks but there are characteristics and dynamics those are unique to particular wireless environment. For example, in the LTE the base station maintains queues per radio bearer per user hence it gives different interaction when all traffic from user share the same queue. Again, the user mobility in a cellular network is different than the user mobility in a Wi-Fi network. Thus, It is important to evaluate the performance of the proposed RMCAT candidates separately in the cellular mobile networks and Wi-Fi local networks (IEEE 802.11xx protocol family).

RMCAT evaluation criteria [I-D.ietf-rmcat-eval-criteria] document provides the guideline to perform the evaluation on candidate algorithms and recognizes wireless networks to be important access link. However, it does not provides particular test cases to evaluate the performance of the candidate algorithm. In this document we describe test cases specifically targeting cellular networks such as LTE networks and Wi-Fi local networks.

2. Terminologies

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

3. Cellular Network Specific Test Cases

A cellular environment is more complicated than a wireline ditto since it seeks to provide services in the context of variable available bandwidth, location dependencies and user mobilities at different speeds. In a cellular network the user may reach the cell edge which may lead to a significant amount of retransmissions to deliver the data from the base station to the destination and vice versa. These network links or radio links will often act as a bottleneck for the rest of the network which will eventually lead to excessive delays or packet drops. An efficient retransmission or

link adaptation mechanism can reduce the packet loss probability but there will still be some packet losses and delay variations. Moreover, with increased cell load or handover to a congested cell, congestion in transport network will become even worse. Besides, there are certain characteristics which make the cellular network different and challenging than other types of access network such as Wi-Fi and wired network. In a cellular network -

- The bottleneck is often a shared link with relatively few users.
  - The cost per bit over the shared link varies over time and is different for different users.
  - Left over/ unused resource can be grabbed by other greedy users.
- Queues are always per radio bearer hence each user can have many of such queues.
- Users can experience both Inter and Intra Radio Access Technology (RAT) handovers ("handover" definition in [HO-def-3GPP]).
- Handover between cells, or change of serving cells (see in [HO-LTE-3GPP] and [HO-UMTS-3GPP]) might cause user plane interruptions which can lead to bursts of packet losses, delay and/or jitter. The exact behavior depends on the type of radio bearer. Typically, the default best effort bearers do not generate packet loss, instead packets are queued up and transmitted once the handover is completed.
- The network part decides how much the user can transmit.
- The cellular network has variable link capacity per user
  - Can vary as fast as a period of milliseconds.
  - Depends on lots of facts (such as distance, speed, interference, different flows).
  - Uses complex and smart link adaptation which makes the link behavior ever more dynamic.
  - The scheduling priority depends on the estimated throughput.
- Both Quality of Service (QoS) and non-QoS radio bearers can be used.

Hence, a real-time communication application operating in such a cellular network need to cope with shared bottleneck link and variable link capacity, event likes handover, non-congestion related loss, abrupt change in bandwidth (both short term and long term) due to handover, network load and bad radio coverage. Even though 3GPP define QoS bearers [QoS-3GPP] to ensure high quality user experience, adaptive real-time applications are desired.

Different mobile operators deploy their own cellular network with their own set of network functionalities and policies. Usually, a mobile operator network includes 2G, EDGE, 3G and 4G radio access technologies. Looking at the specifications of such radio technologies it is evident that only 3G and 4G radio technologies can support the high bandwidth requirements from real-time interactive video applications. The future real-time interactive application will impose even greater demand on cellular network performance which makes 4G (and beyond radio technologies) more suitable access technology for such genre of application.

The key factors to define test cases for cellular network are

- Shared and varying link capacity
- Mobility
- Handover

However, for cellular network it is very hard to separate such events from one another as these events are heavily related. Hence instead of devising separate test cases for all those important events we have divided the test case in two categories. It should be noted that in the following test cases the goal is to evaluate the performance of candidate algorithms over radio interface of the cellular network. Hence it is assumed that the radio interface is the bottleneck link between the communicating peers and that the core network does not add any extra congestion in the path. Also the combination of multiple access technologies such as one user has LTE connection and another has Wi-Fi connection is kept out of the scope of this document. However, later those additional scenarios can also be added in this list of test cases. While defining the test cases we assumed a typical real-time telephony scenario over cellular networks where one real-time session consists of one voice stream and one video stream. We recommend that an LTE network simulator is used for the test cases defined in this document, for example-NS-3 LTE simulator [LTE-simulator].

3.1. Varying Network Load

The goal of this test is to evaluate the performance of the candidate congestion control algorithm under varying network load. The network load variation is created by adding and removing network users a.k.a. User Equipments (UEs) during the simulation. In this test case, each of the user/UE in the media session is an RMCAT compliant endpoint. The arrival of users follows a Poisson distribution, which is proportional to the length of the call, so that the number of users per cell is kept fairly constant during the evaluation period. At the beginning of the simulation there should be enough amount of time to warm-up the network. This is to avoid running the evaluation in an empty network where network nodes are having empty buffers, low interference at the beginning of the simulation. This network initialization period is therefore excluded from the evaluation period.

This test case also includes user mobility and competing traffic. The competing traffics includes both same kind of flows (with same adaptation algorithms) and different kind of flows (with different service and congestion control). The investigated congestion control algorithms should show maximum possible network utilization and stability in terms of rate variations, lowest possible end to end frame latency, network latency and Packet Loss Rate (PLR) at different cell load level.

3.1.1. Network Connection

Each mobile user is connected to a fixed user. The connection between the mobile user and fixed user consists of a LTE radio access, an Evolved Packet Core (EPC) and an Internet connection. The mobile user is connected to the EPC using LTE radio access technology which is further connected to the Internet. The fixed user is connected to the Internet via wired connection with no bottleneck (practically infinite bandwidth). The Internet and wired connection in this setup does not add any network impairments to the test, it only adds 10ms of one-way transport propagation delay.

The path from the fixed user to mobile user is defines as "Downlink" and the path from mobile user to the fixed user is defined as "Uplink". We assume that only uplink or downlink is congested for the mobile users. Hence, we recommend that the uplink and downlink simulations are run separately.

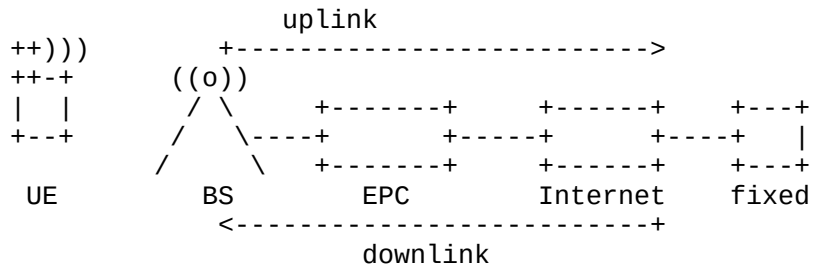


Figure 1: Simulation Topology

3.1.2. Simulation Setup

The values enclosed within " [ ] " for the following simulation attributes follow the notion set in [I-D.ietf-rmcat-eval-test]. The desired simulation setup as follows-

1. Radio environment

   A. Deployment and propagation model : 3GPP case 1[Deployment]

   B. Antenna: Multiple-Input and Multiple-Output (MIMO), [2D, 3D]

   C. Mobility: [3km/h, 30km/h]

   D. Transmission bandwidth: 10Mhz

   E. Number of cells: multi cell deployment (3 Cells per Base Station (BS) * 7 BS) = 21 cells

   F. Cell radius: 166.666 Meters

   G. Scheduler: Proportional fair with no priority

   H. Bearer: Default bearer for all traffic.

   I. Active Queue Management (AQM) settings: AQM [on,off]

2. End to end Round Trip Time (RTT): [40, 150]

3. User arrival model: Poisson arrival model

4. User intensity:

   * Downlink user intensity: {0.7, 1.4, 2.1, 2.8, 3.5, 4.2, 4.9, 5.6, 6.3, 7.0, 7.7, 8.4, 9,1, 9.8, 10.5}

* Uplink user intercity : {0.7, 1.4, 2.1, 2.8, 3.5, 4.2, 4.9, 5.6, 6.3, 7.0}

5. Simulation duration: 91s

6. Evaluation period : 30s-60s

7. Media traffic

    1. Media type: Video

        a. Media direction: [Uplink, Downlink]

        b. Number of Media source per user: One (1)

        c. Media duration per user: 30s

        d. Media source: same as define in section 4.3 of [I-D.ietf-rmcat-eval-test]

    2. Media Type : Audio

        a. Media direction: Uplink and Downlink

        b. Number of Media source per user: One (1)

        c. Media duration per user: 30s

        d. Media codec: Constant BitRate (CBR)

        e. Media bitrate : 20 Kbps

        f. Adaptation: off

8. Other traffic model:

    * Downlink simulation: Maximum of 4Mbps/cell (web browsing or FTP traffic)

    * Unlink simulation: Maximum of 2Mbps/cell (web browsing or FTP traffic)

3.2. Bad Radio Coverage

The goal of this test is to evaluate the performance of candidate congestion control algorithm when users visit part of the network with bad radio coverage. The scenario is created by using larger cell radius than previous test case. In this test case each of the

user/UE in the media session is an RMCAT compliant endpoint. The arrival of users follows a Poisson distribution, which is proportional to the length of the call, so that the number of users per cell is kept fairly constant during the evaluation period. At the beginning of the simulation there should be enough amount of time to warm-up the network. This is to avoid running the evaluation in an empty network where network nodes are having empty buffers, low interference at the beginning of the simulation. This network initialization period is therefore excluded from the evaluation period.

This test case also includes user mobility and competing traffic. The competing traffics includes same kind of flows (with same adaptation algorithms) . The investigated congestion control algorithms should show maximum possible network utilization and stability in terms of rate variations, lowest possible end to end frame latency, network latency and Packet Loss Rate (PLR) at different cell load level.

3.2.1. Network connection

Same as defined in Section 3.1.1

3.2.2. Simulation Setup

The desired simulation setup is same as Varying Network Load test case defined in Section 3.1 except following changes-

1. Radio environment : Same as defined in Section 3.1.2 except followings

   A. Deployment and propagation model : 3GPP case 3[Deployment]

   B. Cell radius: 577.3333 Meters

   C. Mobility: 3km/h

2. User intensity = {0.7, 1.4, 2.1, 2.8, 3.5, 4.2, 4.9, 5.6, 6.3, 7.0}

3. Media traffic model: Same as defined in Section 3.1.2

4. Other traffic model: None

3.3. Desired Evaluation Metrics for cellular test cases

RMCAT evaluation criteria document [I-D.ietf-rmcat-eval-criteria] defines metrics to be used to evaluate candidate algorithms. However, looking at the nature and distinction of cellular networks we recommend at minimum following metrics to be used to evaluate the performance of the candidate algorithms for the test cases defined in this document.

The desired metrics are-

- Average cell throughput (for all cells), shows cell utilizations.
- Application sending and receiving bitrate, goodput.
- Packet Loss Rate (PLR).
- End to end Media frame delay. For video, this means the delay from capture to display.
- Transport delay.
- Algorithm stability in terms of rate variation.

4. Wi-Fi Networks Specific Test Cases

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

- The wireless radio channel is subject to interference from nearby transmitters, multipath fading, and shadowing, causing fluctuations in link throughput and sometimes an error-prone communication environment
- Available network bandwidth is not only shared over the air between cocurrent users, but also between uplink and downlink traffic due to the half duplex nature of wireless transmission medium.
- Packet transmessions over Wi-Fi are susceptible to contentions and collisions over the air. Consequently, traffic load beyond a certain utilization level over a Wi-Fi network can introduce frequent collisions and significant network overhead. This, in turn, leads to excessive delay, retransmission, loss and lower effective bandwidth for applications.

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

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

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

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

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

Throughout this draft, unless otherwise mentioned, test cases are described using 802.11n due to its wide availability in real-world networks. Statistics collected from enterprise Wi-Fi networks show that the dominant physical modes are 802.11n and 802.11ac, accounting for 73.6% and 22.5% of enterprise network users, respectively.

Since Wi-Fi network normally connects to a wired infrastructure, either the wired network or the Wi-Fi network could be the bottleneck. In the following section, we describe basic test cases for both scenarios separately. The same set of performance metrics as in [I-D.ietf-rmcat-eval-test]) should be collected for each test case.

While all test cases described below can be carried out using simulations, e.g. based on [ns-2] or [ns-3], it is also recommended to perform testbed-based evaluations using Wi-Fi access points and endpoints running up-to-date IEEE 802.11 protocols. [Editor's Note: need to add some more discussions on the pros and cons of simulation-based vs. testbed-based evaluations. Will be good to provide recommended testbed configurations.]

4.1. Bottleneck in Wired Network

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

4.1.1. Network topology

Figure 2 shows topology of the network for Wi-Fi test cases. The test contains multiple mobile nodes (MNs) connected to a common Wi-Fi access point (AP) and their corresponding wired clients on fixed nodes (FNs). Each connection carries either RMCAT or TCP traffic flow. Directions of the flows can be uplink, downlink, or bi-directional.

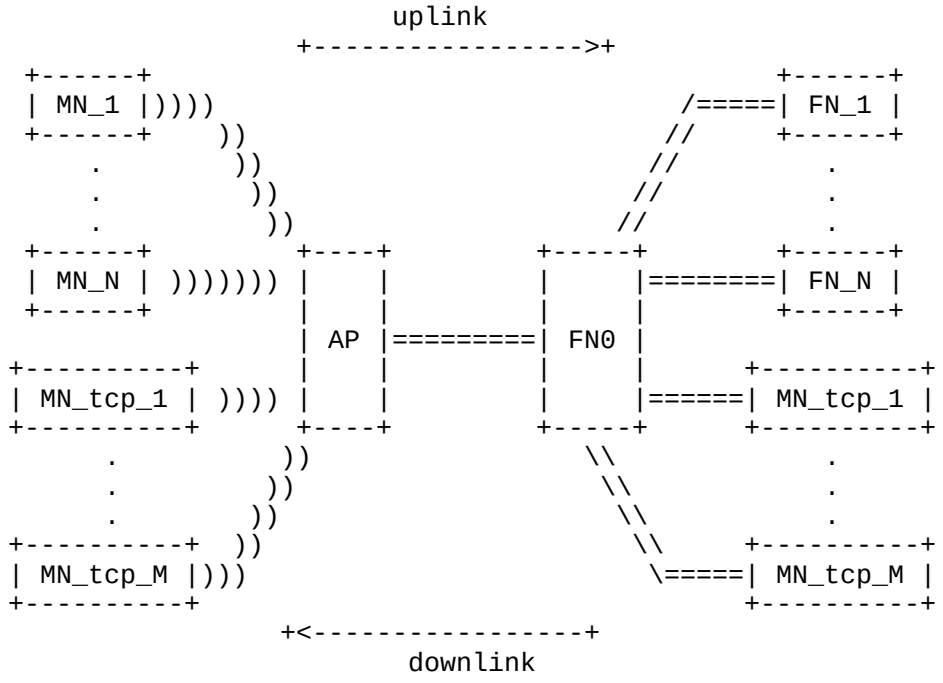


Figure 2: Network topology for Wi-Fi test cases

4.1.2. Test setup

- Test duration: 120s
- Wi-Fi network characteristics:
  - Radio propagation model: Log-distance path loss propagation model [NS3WiFi]
  - PHY- and MAC-layer configuration: IEEE 802.11n
  - MCS Index at 11: 16-QAM 1/2, Raw Data Rate@52Mbps
- Wired path characteristics:
  - Path capacity: 1Mbps
  - One-Way propagation delay: 50ms.
  - Maximum end-to-end jitter: 30ms
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue size: 300ms.
  - Path loss ratio: 0%.
- Application characteristics:
  - Media Traffic:
    - Media type: Video
    - Media direction: See Section 4.1.3
    - Number of media sources (N): See Section 4.1.3
    - Media timeline:
      - Start time: 0s.
      - End time: 119s.
  - Competing traffic:
    - Type of sources: long-lived TCP or CBR over UDP
    - Traffic direction: See Section 4.1.3

+ Number of sources (M): See Section 4.1.3

+ Congestion control: Default TCP congestion control [TBD] or CBR over UDP

+ Traffic timeline: See Section 4.1.3

4.1.3. Typical test scenarios

- Single uplink RMCAT flow: N=1 with uplink direction and M=0.

- One pair of bi-directional RMCAT flows: N=2 (with one uplink flow and one downlink flow); M=0.

- One pair of bi-directional RMCAT flows, one on-off CBR over UDP flow on uplink : N=2 (with one uplink flow and one downlink flow); M=1 (uplink). CBR flow on time at 0s-60s, off time at 60s-119s

- One pair of bi-directional RMCAT flows, one off-on CBR over UDP flow on uplink : N=2 (with one uplink flow and one downlink flow); M=1 (uplink). UDP off time: 0s-60s, on time: 60s-119s

- One RMCAT flow competing against one long-live TCP flow over uplink: N=1 (uplink) and M = 1(uplink), TCP start time: 0s, end time: 119s.

4.1.4. Expected behavior

- Single uplink RMCAT flow: the candidate algorithm is expected to detect the path capacity constraint, converges to bottleneck link’s capacity and adapt the flow to avoid unwanted oscillation when the sending bit rate is approaching the bottleneck link’s capacity. No excessivie rate oscillations.

- Bi-directional RMCAT flows: It is expected that the candidate algorithms is able to converge to the bottleneck capacity of the wired path on both directions despite presense of measurment noise over the Wi-Fi connection. In the presence of background TCP or CBR over UDP traffic, the rate of RMCAT flows should adapt in a timely manner to changes in the available bottleneck bandwidth.

- One RMCAT flow competing with long-live TCP flow over uplink: the candidate algorithm should be able to avoid congestion collapse, and stablize at a fair share of the bottleneck capacity over the wired path.

4.2. Bottleneck in Wi-Fi Network

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

4.2.1. Network topology

Same as defined in Section 4.1.1

4.2.2. Test setup

- Test duration: 120s
- Wi-Fi network characteristics:
  - Radio propagation model: Log-distance path loss propagation model [NS3WiFi]
  - PHY- and MAC-layer configuration: IEEE 802.11n
  - MCS Index at 11: 16-QAM 1/2, Raw Data Rate at 52Mbps
- Wired path characteristics:
  - Path capacity: 100Mbps
  - One-Way propagation delay: 50ms.
  - Maximum end-to-end jitter: 30ms
  - Bottleneck queue type: Drop tail.
  - Bottleneck queue size: 300ms.
  - Path loss ratio: 0%.
- Application characteristics:
  - Media Traffic:
    - Media type: Video
    - Media direction: See Section 4.2.3
    - Number of media sources (N): See Section 4.2.3
    - Media timeline:

- Start time: 0s.

- End time: 119s.

* Competing traffic:

  + Type of sources: long-lived TCP or CBR over UDP

  + Number of sources (M): See Section 4.2.3

  + Traffic direction: See Section 4.2.3

  + Congestion control: Default TCP congestion control [TBD] or CBR over UDP

  + Traffic timeline: See Section 4.2.3

4.2.3. Typical test scenarios

This sections describes a few specific test scenarios that are deemed as important for understanding behavior of a RMCAT candidate solution over a Wi-Fi network.

- Multiple RMCAT Flows Sharing the Wireless Downlink: N=16 (all downlink); M = 0; This test case is for studying the impact of contention on competing RMCAT flows. Specifications for IEEE 802.11n, MCS Index at 11: 16-QAM 1/2, Raw Data Rate at 52Mbps is chosen. Note that retransmissions, MAC-layer headers, and control packets may be sent at a lower link speed. The total application-layer throughput (reasonable distance, low interference and small number of contention stations) for 802.11n is around 20 Mbps. Consequently, a total of N=16 RMCAT flows are needed for saturating the wireless interface in this experiment. Evaluation of a given candidate solution should focus on whether downlink RMCAT flows can stablize at a fair share of bandwidth.

- Multiple RMCAT Flows Sharing the Wireless Uplink: N = 16 (all downlink); M = 0; When multiple clients attempt to transmit video packets uplink over the wireless interface, they introduce more frequent contentions and potentially collisions. Per-flow throughput is expected to be lower than that in the previous downlink-only scenario. Evaluation of a given candidate solution should focus on whether uplink flows can stablize at a fair share of bandwidth.

- Multiple Bi-directional RMCAT Flows: N = 16 (8 uplink and 8 downlink); M = 0. the goal of this test is to evaluate

performance of the candidate solution in terms of bandwidth fairness between uplink and downlink flow.

- Multiple Bi-directional RMCAT Flows with on-off CBR traffic: N = 16 (8 uplink and 8 downlink); M = 5(uplink). The goal of this test is to evaluate upgrading performance of the candidate solution in terms of available bandwidth changes caused by the CBR uplink flow over UDP. CBR over UDP background flows have on time 0s-60s, and off time 60s-119s

- Multiple Bi-directional RMCAT Flows with off-on CBR traffic: N = 16 (8 uplink and 8 downlink); M = 5(uplink). The goal of this test is to evaluate upgrading performance of the candidate solution in terms of available bandwidth changes caused by the CBR uplink flow over UDP. CBR over UDP background flows have off time 0s-60s, and on time 60s-119s.

- Multiple RMCAT flows in the presence of background TCP traffic: the goal of this test is to evaluate how RMCAT flows compete against TCP over a congested Wi-Fi network for a given candidate solution. TCP start time: 0s, end time: 119s. [Editor’s Note: more detailed description will be added in the next version in terms of directoin/number of RMCAT and TCP flows.]

- Varying number of RMCAT flows: the goal of this test is to evaluate how a candidate RMCAT solution responds to varying traffic load/demand over a congested Wi-Fi network. [Editor’s Note: more detailed description will be added in the next version in terms of arrival/departure pattern of the flows.]

4.2.4. Expected behavior

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

- Multiple uplink RMCAT flows: overall bandwidth usage shared by all RMCAT flows should be no less than those shared by the same number of TCP flows (i.e., benchmark performance using TCP flows).

- Multiple bi-directional RMCAT flows with CBR over UDP traffic: RMCAT flows should adapt to the changes in available bandwidth.

- Multiple bi-directional RMCAT flows with TCP traffic: overall bandwidth usage shared by all RMCAT flows should be no less than those shared by the same number of TCP flows (i.e., benchmark

performance using TCP flows). All downlink RMCAT flows are expected to obtain similar bandwidth with respect to each other.

4.3. Potential Potential Test Cases

4.3.1. EDCA/WMM usage

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

4.3.2. Legacy 802.11b Effects

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

5. Conclusion

This document defines a collection of test cases that are considered important for cellular and Wi-Fi networks. Moreover, this document also provides a framework for defining additional test cases over wireless cellular/Wi-Fi networks.

6. Acknowledgements

We would like to thank Tomas Frankkila, Magnus Westerlund, Kristofer Sandlund for their valuable comments while writing this draft.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

Security issues have not been discussed in this memo.

9. References

9.1. Normative References

[Deployment]
TS 25.814, 3GPP., "Physical layer aspects for evolved Universal Terrestrial Radio Access (UTRA)", October 2006, <http://www.3gpp.org/ftp/specs/archive/25_series/25.814/25814-710.zip>.

[HO-def-3GPP]
TR 21.905, 3GPP., "Vocabulary for 3GPP Specifications", December 2009, <http://www.3gpp.org/ftp/specs/archive/21_series/21.905/21905-940.zip>.

[HO-LTE-3GPP]
TS 36.331, 3GPP., "E-UTRA- Radio Resource Control (RRC); Protocol specification", December 2011, <http://www.3gpp.org/ftp/specs/archive/36_series/36.331/36331-990.zip>.

[HO-UMTS-3GPP]
TS 25.331, 3GPP., "Radio Resource Control (RRC); Protocol specification", December 2011, <http://www.3gpp.org/ftp/specs/archive/25_series/25.331/25331-990.zip>.

[I-D.ietf-rmcat-eval-criteria]
Varun, V., Ott, J., and S. Holmer, "Evaluating Congestion Control for Interactive Real-time Media", draft-ietf-rmcat-eval-criteria-05 (work in progress), March 2016.

[NS3WiFi] "Wi-Fi Channel Model in NS3 Simulator", <https://www.nsnam.org/doxygen/classns3_1_1_yans_wifi_channel.html>.

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

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

9.2. Informative References

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

[I-D.ietf-rmcat-eval-test]
Sarker, Z., Varun, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-ietf-rmcat-eval-test-03 (work in progress), March 2016.

[IEEE802.11]
"Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", 2012.

[LTE-simulator]
"NS-3, A discrete-Event Network Simulator", <https://www.nsnam.org/docs/release/3.23/manual/html/index.html>.

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

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

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Laboratoriegraend 11
Luleae 97753
Sweden

Phone: +46 107173743
Email: zaheduzzaman.sarker@ericsson.com

Ingemar Johansson
Ericsson AB
Laboratoriegraend 11
Luleae 97753
Sweden

Phone: +46 10 7143042
Email: ingemar.s.johansson@ericsson.com

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

Email: xiaoqzhu@cisco.com

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

Email: jianfu@cisco.com

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

Email: dtan2@cisco.com

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

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

RMCAT WG
Internet-Draft
Intended status: Informational
Expires: January 8, 2017

X. Zhu
Cisco Systems
Z. Sarker
Ericsson AB

July 7, 2016

Framework for Real-time Media Congestion Avoidance Techniques
draft-zhu-rmcat-framework-00

Abstract

Congestion control is an essential element in ensuring fair bandwidth usage and preventing congestion collapse for traffic sharing the Internet. For interactive real-time media traffic such as video conferencing, design of congestion control solution also needs to account for many other factors such as the requirement for low latency packet delivery and interactions with live video encoder. This document describes a common framework with the core functional building blocks for a real-time media congestion solutions.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 2017.

Copyright Notice

Table of Contents

1. Introduction

Given increasing amount of interactive real-time media traffic over the Internet, such as video conferencing, it is important that these applications employ proper congestion control mechanisms to avoid congestion collapse. [I-D.ietf-rmcat-cc-requirements] specifies the list of requirements of a viable solution.

This document outlines a common framework for designing a congestion control mechanism for real-time interactive communication, so that individual drafts on specific solutions follow a consistent set of terminologies in describing their respective components. The next section (Section 3) describes common functional modules in this framework, whereas Section 4 provides examples on how these modules build together to support single and multiple media streams.

[Editor's note : This document does not describe the interaction between application, codec and congestion control system. The interaction among application, codec and congestion control system are defined in other documents. There is a possibility to merge all the documents into one single document.]

2. Key Words for Requirements

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

3. Functional Modules

A viable solution for real-time media congestion control needs to comprise of several common modules. This section provides a brief description of them and their respective functionalities. A congestion control solution for real-time media should comprise of the described functional modules. This should help understanding different congestion control solutions.

- Network Congestion Controller : this is the core module for estimating available bandwidth over the network based on periodic RTCP feedback reports [RFC3550] from the receiver. This module contains key functions and calculations required to detect congestion and estimate available bandwidth on the transmission path based on the reception quality of the media traffic. Different congestion control solutions employ different algorithms in detecting congestion and estimating available bandwidth for its media flow. It also possible that multiple media streams are multiplexed over a single transport, hence share a common congestion control module in aggregation.

- Transmission Queue : this module is needed to absorb the instantaneous mismatch between output from a live video encoder and regulated outgoing media flow. The transmission queue schedules outgoing traffic according to sending rate recommended by the rate controller module. It reports back its occupancy level to the rate controller module to assist future rate control decisions on target video rate, sending rate, and probing rate.

- Rate Controller : this module takes the estimated available bandwidth from the network congestion controller, shared states of other flows, as well as occupancy level of the transmission queue as input. It makes holistic decisions on: a) target video rate for the live video encoder; b) sending rate for regulating outgoing media flow(s) for the transmission queue; and c) rate of probing packets when needed. In the case where multiple media streams share a single transport and a common network congestion controller (for estimating available bandwidth in aggregation), the rate controller is also responsible for distributing available bandwidth amongst different media streams according to their relative priorities as well as share state information. When losses occur over the network and some previous media packets need to be retransmitted, the rate controller should also account for the bandwidth needed for retransmission.

- Network Probe Generator: A congestion control solution can actively probe to estimate the available bandwidth on the media transmission path by sending more than what the live video encoder

produces. Such an approach can be especially effective during the ramp up period of media and transmission rates, when no congestion has been observed over the network yet. The network probe generator is responsible for generating probing packets according to the probing rate specified by the rate controller. It can employ different techniques in doing so -- for example by generating simple dummy packets with unknown payload type or by generating Forward Error Correction (FEC) packets. While this document does not specify what probing technique to use or how those packets should be generated, a complete congestion control solution needs should specify total rate of the probe packets via the rate controller module.

- o Live Video Encoder : the sender typically also contains a live video encoder, which adjusts the its encoding parameters according to the target video rate set by the rate controller. The output rate from the video encoder may deviate from this target due to uncertainty in the captured video content characteristics and the encoder rate control process. The output encoded media packets are fed to the transmission queue. Note that internal operations of the live video encoder (i.e., how video encoder rate control works) is out of scope for this document.

- o Shared State: In the case of multiple media streams sharing a common sender hence a common network congestion controller, the sender should also contain a shared state module for storage and exchange of congestion control states [Editor’s Note from Xiaoqing: examples of congestion control states??] amongst the multiple flows.

4. Example Configurations

4.1. Example Configurations for a Single Stream

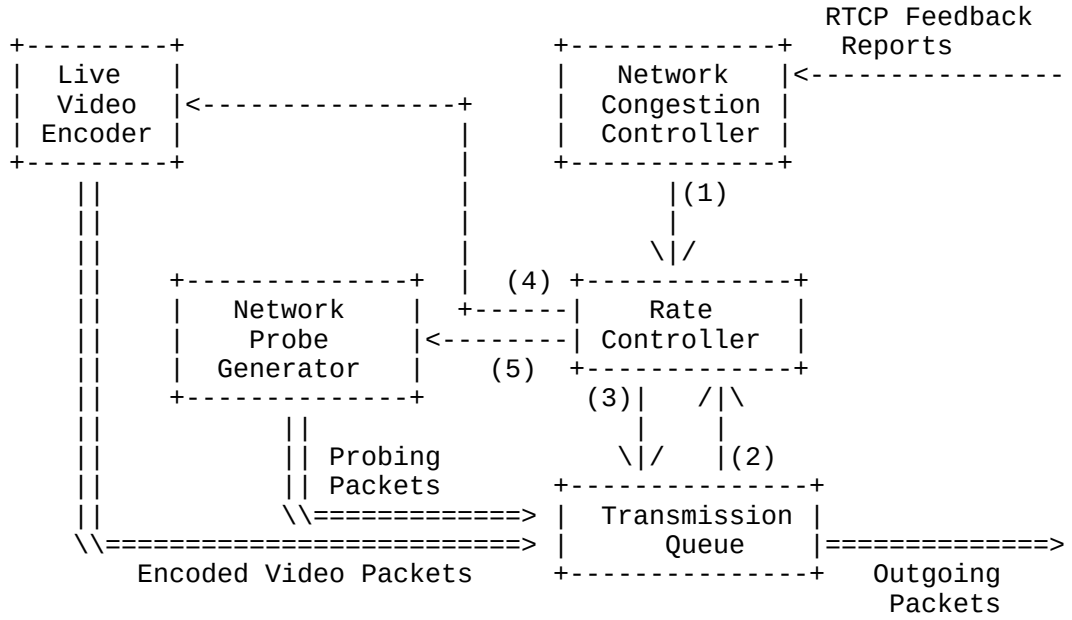


Figure 1: RMCAT Solution Framework at the Sender: Single Stream

Figure 1 shows an example configuration at the sender for supporting a single media stream. The Network Congestion Controller estimates available bandwidth based on periodic RTCP feedback reports. The Rate Controller takes as input the estimated available bandwidth (1) and the current occupancy level of the Transmission Queue (2). It calculates as output sending rate (3) for the Transmission Queue, video target rate (4) for the Live Video Encoder, and probing rate (5) -- if they are needed -- for the Network Probe Generator. The Transmission Queue holds packets generated by both the Live Video Encoder and the Network Probe Generator; it paces transmission of its outgoing packets according to the sending rate (3) specified by Rate Controller.

Obviously, it is possible for a congestion control solution to contain alternative configurations between these functional modules. [TODO: add one quick example on alternative wiring.] It is required that the candidate solution draft specify how their internal functional modules align to this framework.

4.2. Example Configurations for Multiple Streams

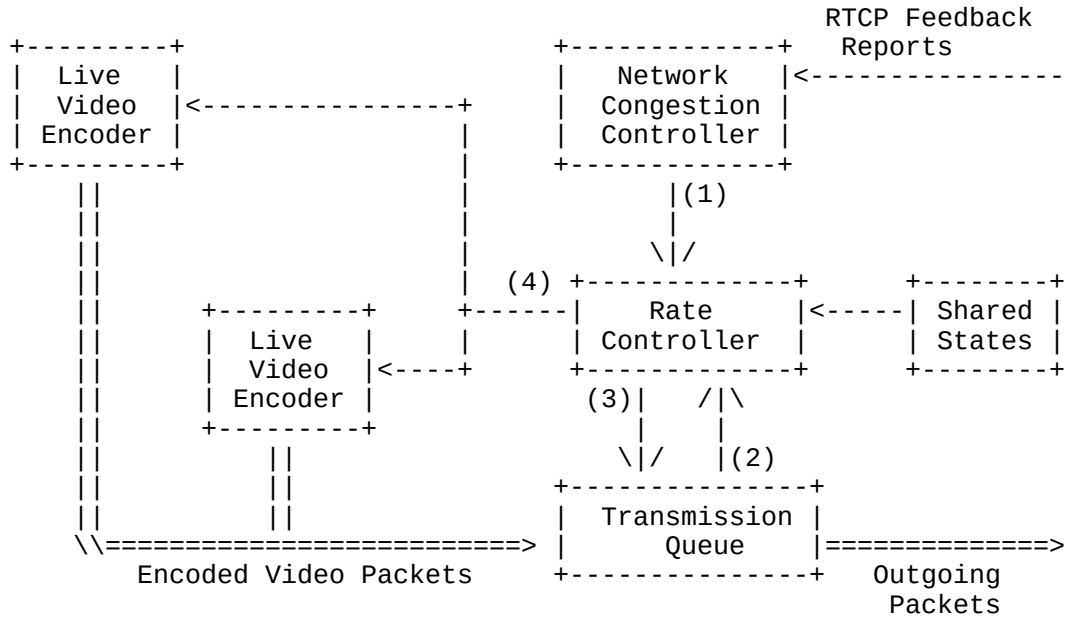


Figure 2: RMCAT Solution Framework at the Sender: Multiple Streams

Figure 2 shows an example configuration for multiple video streams sharing a common Network Congestion Controller. The Network Congestion Controller calculates an aggregated estimated available bandwidth (1) based on periodic RTCP feedback reports. The Rate Controller divides up the aggregate estimated bandwidth (1) from the Network Congestion Controller amongst sub-streams based on their relative priority levels, Shared States, as well as current occupancy level of the Transmission Queue. It subsequently determines the per-flow sending rate (3) as regulated by the Transmission Queue and target video rate (4) for each flow.

In this specific example, the transmission queue is envisioned as a logical entity. For instance, this transmission queue can be implemented priority-based scheduling and one physical queue per stream. For sake of simplicity the role of Network Probe Generator is omitted in the above figure.

5. Acknowledgements

The RMCAT design team discussions contributed to this memo.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

TBD

8. References

8.1. Normative References

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

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

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

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <http://www.rfc-editor.org/info/rfc3551>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <http://www.rfc-editor.org/info/rfc5104>.

8.2. Informative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

Authors' Addresses

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

Email: xiaoqzhu@cisco.com

Zaheduzzaman Sarker
Ericsson AB
Luleae
Sweden

Phone: 00 46 10 717 37 43
Email: zaheduzzaman.sarker@ericsson.com