

RTP Media Congestion Avoidance
Techniques (rmcat)
Internet-Draft
Intended status: Experimental
Expires: March 17, 2016

S. Islam
M. Welzl
S. Gjessing
University of Oslo
September 14, 2015

Coupled congestion control for RTP media
draft-ietf-rmcat-coupled-cc-00

Abstract

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

Status of this Memo

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

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

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

This Internet-Draft will expire on March 17, 2016.

Copyright Notice

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

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

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

Table of Contents

1. Introduction	3
2. Definitions	3
3. Limitations	4
4. Architectural overview	5
5. Roles	6
5.1. SBD	6
5.2. FSE	6
5.3. Flows	7
5.3.1. Example algorithm 1 - Active FSE	7
5.3.2. Example algorithm 2 - Conservative Active FSE	8
6. Application	10
6.1. NADA	10
6.2. General recommendations	10
7. Acknowledgements	11
8. IANA Considerations	11
9. Security Considerations	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Appendix A. Scheduling	13
Appendix B. Example algorithm - Passive FSE	13
B.1. Example operation (passive)	15
Authors' Addresses	20

1. Introduction

When there is enough data to send, a congestion controller must increase its sending rate until the path's capacity has been reached; depending on the controller, sometimes the rate is increased further, until packets are ECN-marked or dropped. This process inevitably creates undesirable queuing delay -- an effect that is amplified when multiple congestion controlled connections traverse the same network bottleneck.

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 [rtcweb-usecases].

2. Definitions

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

Available Bandwidth:

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

Bottleneck:

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

Flow:

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

Flow Group Identifier (FGI):

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

Flow State Exchange (FSE):

The entity that maintains information that is exchanged between flows.

Flow Group (FG):

A group of flows having the same FGI.

Shared Bottleneck Detection (SBD):

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

3. Limitations

Sender-side only:

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

Shared bottlenecks do not change quickly:

As per the definition above, a bottleneck depends on cross traffic, and since such traffic can heavily fluctuate, bottlenecks can change at a high frequency (e.g., there can be oscillation between two or more links). This means that, when flows are partially routed along different paths, they may quickly change between sharing and not sharing a bottleneck. For simplicity, here it is assumed that a shared bottleneck is valid for a time interval that is significantly longer than the interval at which congestion controllers operate. Note that, for the only SBD mechanism defined in this document (multiplexing on the same five-tuple), the notion of a shared bottleneck stays correct even in the presence of fast traffic fluctuations: since all flows that are assumed to share a bottleneck are routed in the same way, if the bottleneck changes, it will still be shared.

4. Architectural overview

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

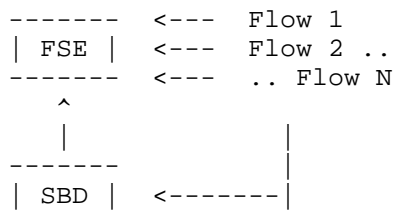


Figure 1: Coupled congestion control architecture

Since everything shown in Figure 1 is assumed to operate on a single host (the sender) only, this document only describes aspects that have an influence on the resulting on-the-wire behavior. It does, for instance, not define how many bits must be used to represent FSIs, or in which way the entities communicate. Implementations can take various forms: for instance, all the elements in the figure could be implemented within a single application, thereby operating on flows generated by that application only. Another alternative could be to implement both the FSE and SBD together in a separate process which different applications communicate with via some form of Inter-Process Communication (IPC). Such an implementation would extend the scope to flows generated by multiple applications. The FSE and SBD could also be included in the Operating System kernel.

5. Roles

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

5.1. SBD

SBD uses knowledge about the flows to determine which flows belong in the same Flow Group (FG), and assigns FGIs accordingly. This knowledge can be derived in three basic ways:

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

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

5.2. FSE

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

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

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

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

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

5.3. Flows

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

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

5.3.1. Example algorithm 1 - Active FSE

This algorithm was designed to be the simplest possible method to assign rates according to the priorities of flows. Simulations

results in [fse] indicate that it does however not significantly reduce queuing delay and packet loss.

- (1) When a flow f starts, it registers itself with SBD and the FSE. FSE_R is initialized with the congestion controller's initial rate. SBD will assign the correct FGI. When a flow is assigned an FGI, it adds its FSE_R to S_CR.
- (2) When a flow f stops, its entry is removed from the list.
- (3) Every time the congestion controller of the flow f determines a new sending rate CC_R, the flow calls UPDATE, which carries out the tasks listed below to derive the new sending rates for all the flows in the FG. A flow's UPDATE function uses a local (i.e. per-flow) temporary variable S_P, which is the sum of all the priorities.
 - (a) It updates S_CR.

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

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

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

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

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

5.3.2. Example algorithm 2 - Conservative Active FSE

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

- (1) When a flow *f* starts, it registers itself with SBD and the FSE. FSE_R is initialized with the congestion controller's initial rate. SBD will assign the correct FGI. When a flow is assigned an FGI, it adds its FSE_R to S_CR.
- (2) When a flow *f* stops, its entry is removed from the list.
- (3) Every time the congestion controller of the flow *f* determines a new sending rate CC_R, the flow calls UPDATE, which carries out the tasks listed below to derive the new sending rates for all the flows in the FG. A flow's UPDATE function uses a local (i.e. per-flow) temporary variable S_P, which is the sum of all the priorities, and a local variable DELTA, which is used to calculate the difference between CC_R and the previously stored FSE_R. To prevent flows from either ignoring congestion or overreacting, a timer keeps them from changing their rates immediately after the common rate reduction that follows a congestion event. This timer is set to 2 RTTs of the flow that experienced congestion because it is assumed that a congestion event can persist for up to one RTT of that flow, with another RTT added to compensate for fluctuations in the measured RTT value.

- (a) It updates S_CR based on DELTA.

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

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

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

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

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

6. Application

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

6.1. NADA

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

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

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

6.2. General recommendations

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

Receiver-side calculations:

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

7. Acknowledgements

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

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

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

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

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

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, 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); extended version available as a technical report from <http://safiquili.at.ifi.uio.no/paper/fse-tech-report.pdf> , 2014.

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

[rtcweb-rtp-usage] Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-18.txt (work in progress), October 2014.

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

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

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

Appendix A. 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.

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

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

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

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

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

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

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

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

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

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

(d) It calculates the sending rate, Rate.

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

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

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

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

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

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

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_R(f) = 8. DELTA is negative, hence S_CR = 9;
    DR(f) = 8.
3 c) S_P = 1.5.
3 d) new sending rate = min(infinity, 1/1.5 * 9 + 0) = 6.
3 e) FSE_R(f) = 6.
```

The resulting FSE looks as follows:

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

S_CR = 9, TLO = 0

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

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

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

RTP Media Congestion Avoidance Techniques (rmcat)
Internet-Draft

Intended status: Experimental

Expires: February 23, 2020

S. Islam
M. Welzl
S. Gjessing
University of Oslo
August 22, 2019

Coupled congestion control for RTP media
draft-ietf-rmcat-coupled-cc-09

Abstract

When multiple congestion controlled Real-time Transport Protocol (RTP) sessions traverse the same network bottleneck, combining their controls can improve the total on-the-wire behavior in terms of delay, loss and fairness. This document describes such a method for flows that have the same sender, in a way that is as flexible and simple as possible while minimizing the amount of changes needed to existing RTP applications. It specifies how to apply the method for the Network-Assisted Dynamic Adaptation (NADA) congestion control algorithm, and provides suggestions on how to apply it to other 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 <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 February 23, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Definitions	3
3. Limitations	4
4. Architectural overview	5
5. Roles	6
5.1. SBD	6
5.2. FSE	7
5.3. Flows	8
5.3.1. Example algorithm 1 - Active FSE	9
5.3.2. Example algorithm 2 - Conservative Active FSE	10
6. Application	11
6.1. NADA	11
6.2. General recommendations	11
7. Expected feedback from experiments	12
8. Acknowledgements	12
9. IANA Considerations	13
10. Security Considerations	13
11. References	13
11.1. Normative References	13
11.2. Informative References	14
Appendix A. Application to GCC	16
Appendix B. Scheduling	16
Appendix C. Example algorithm - Passive FSE	16
C.1. Example operation (passive)	19
Appendix D. Change log	23
D.1. draft-welzl-rmcat-coupled-cc	23
D.1.1. Changes from -00 to -01	23
D.1.2. Changes from -01 to -02	23
D.1.3. Changes from -02 to -03	23
D.1.4. Changes from -03 to -04	24
D.1.5. Changes from -04 to -05	24
D.2. draft-ietf-rmcat-coupled-cc	24
D.2.1. Changes from draft-welzl-rmcat-coupled-cc-05	24
D.2.2. Changes from -00 to -01	24
D.2.3. Changes from -01 to -02	24
D.2.4. Changes from -02 to -03	24
D.2.5. Changes from -03 to -04	24
D.2.6. Changes from -04 to -05	25
D.2.7. Changes from -05 to -06	25

D.2.8. Changes from -06 to -07	25
D.2.9. Changes from -07 to -08	25
D.2.10. Changes from -08 to -09	25
Authors' Addresses	25

1. Introduction

When there is enough data to send, a congestion controller attempts to increase its sending rate until the path's capacity has been reached. Some controllers detect path capacity by increasing the sending rate further, until packets are ECN-marked [RFC8087] or dropped, and then decreasing the sending rate until that stops happening. This process inevitably creates undesirable queuing delay when multiple congestion-controlled connections traverse the same network bottleneck, and each connection overshoots the path capacity as it determines its sending rate.

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 [I-D.ietf-rtcweb-overview] [RFC7478].

The described mechanisms are believed safe to use, but are experimental and are presented for wider review and operational evaluation.

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, or an RTP stream [RFC7656], whether or not this RTP stream is multiplexed onto an RTP session with other RTP streams.

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:

Shared bottlenecks can exist when multiple flows originate from the same sender, or when flows from different senders reach the same receiver (see [RFC8382], section 3). Coupled congestion control as described here only supports the former case, not the latter, as it operates inside a single host on the sender side.

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 Group Identifiers (FGIs).

This document describes both active and passive versions. While the passive algorithm works better for congestion controls with RTT-independent convergence, it can still produce oscillations on short time scales. The passive algorithm, described in Appendix C, is therefore considered as highly experimental and not safe to deploy outside of testbed environments. Figure 2 shows the interaction between flows and the FSE, using the variable names defined in Section 5.2.

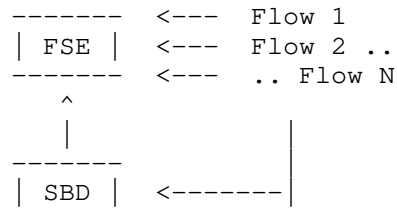


Figure 1: Coupled congestion control architecture

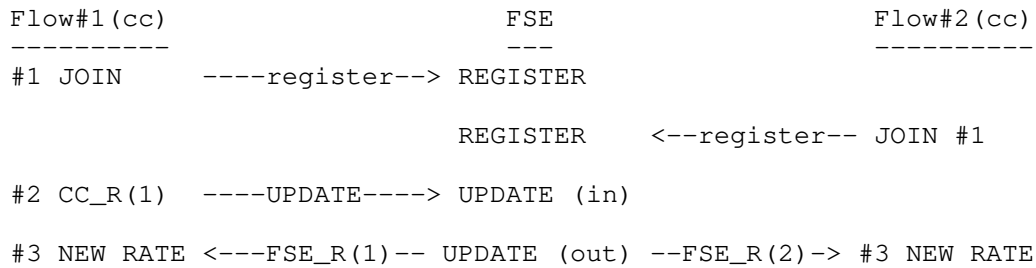


Figure 2: Flow-FSE interaction

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 not, for instance, define how many bits must be used to represent FGIs, 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. However, only one type of coupling algorithm should be used for all flows. Combinations of multiple algorithms at different aggregation levels (e.g., the Operating System coupling application aggregates with one algorithm, and applications coupling their flows with another) have not been tested and are therefore not recommended.

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 values for the Differentiated Services Code Point (DSCP) and the ECN field in the IP header are typically treated in the same way along the path. This method is the only one specified in this document: SBD MAY consider all flows that use the same five-tuple, DSCP and ECN field value 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 [RFC8382]. 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 [RFC8382] for details). Using system configuration to decide about shared bottlenecks can be more efficient (faster to obtain) than using measurements, but it relies on assumptions about the network environment.

5.2. FSE

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

- o a unique flow number f to identify the flow.
- o the FGI of the FG that it belongs to (based on the definitions in this document, a flow has only one bottleneck, and can therefore be in only one FG).
- o a priority $P(f)$, which is a positive number, greater than zero.
- o The rate used by the flow in bits per second, $FSE_R(f)$.

- o The desired rate $DR(f)$ of flow f . This can be smaller than $FSE_R(f)$ if the application feeding into the flow has less data to send than $FSE_R(f)$ would allow, or if a maximum value is imposed on the rate. In the absence of such limits $DR(f)$ must be set to the sending rate provided by the congestion control module of flow f .

Note that the absolute range of priorities does not matter: the algorithm works with a flow's priority portion of the sum of all priority values. For example, if there are two flows, flow 1 with priority 1 and flow 2 with priority 2, the sum of the priorities is 3. Then, flow 1 will be assigned 1/3 of the aggregate sending rate and flow 2 will be assigned 2/3 of the aggregate sending rate. Priorities can be mapped to the "very-low", "low", "medium" or "high" priority levels described in [I-D.ietf-rtcweb-transports] by simply using the values 1, 2, 4 and 8, respectively.

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. Names of variables used in the algorithms are explained below.

- o $CC_R(f)$ - The rate received from the congestion controller of flow f when it calls UPDATE.
- o $FSE_R(f)$ - The rate calculated by the FSE for flow f .
- o $DR(f)$ - The desired rate of flow f .

- o S_CR - The sum of the calculated rates of all flows in the same FG; this value is used to calculate the sending rate.
- o FG - A group of flows having the same FGI, and hence sharing the same bottleneck.
- o P(f) - The priority of flow f which is received from the flow's congestion controller; the FSE uses this variable for calculating FSE_R(f).
- o S_P - The sum of all the priorities.
- o TLO - The total leftover rate: the sum of rates that could not be assigned to flows that were limited by their desired rate.
- o AR - The aggregate rate that is assigned to flows that are not limited by their desired rate.

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(f) 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(f) to S_CR.
- (2) When a flow f stops or pauses, its entry is removed from the list.
- (3) Every time the congestion controller of the flow f determines a new sending rate CC_R(f), 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 three local (i.e. per-flow) temporary variables: S_P, TLO and AR.

- (a) It updates S_CR.

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

- (b) It calculates the sum of all the priorities, S_P, and initializes FSE_R.

```

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

```

- (c) It distributes S_{CR} among all flows, ensuring that each flow's desired rate is not exceeded.

```

TLO = S_CR
while(TLO-AR>0 and S_P>0)
  AR = 0
  for all flows i in FG do
    if FSE_R[i] < DR[i] then
      if TLO * P[i] / S_P >= DR[i] then
        TLO = TLO - DR[i]
        FSE_R[i] = DR[i]
        S_P = S_P - P[i]
      else
        FSE_R[i] = TLO * P[i] / S_P
        AR = AR + TLO * P[i] / S_P
      end if
    end if
  end for
end while

```

- (d) It distributes FSE_R to all the flows.

```

for all flows i in FG do
  send FSE_R(i) to the flow i
end for

```

5.3.2. Example algorithm 2 - Conservative Active FSE

This algorithm changes 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.

Step (a) of the UPDATE function is changed as described below. This also introduces a local variable Δ , which is used to calculate the difference between $CC_R(f)$ and the previously stored $FSE_R(f)$. 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 was not set then
  DELTA = CC_R(f) - FSE_R(f)
  if DELTA < 0 then // Reduce S_CR proportionally
    S_CR = S_CR * CC_R(f) / FSE_R(f)
    Set Timer for 2 RTTs
  else
    S_CR = S_CR + DELTA
  end if
end if
```

6. Application

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

6.1. NADA

Network-Assisted Dynamic Adaption (NADA) [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. 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].

Stateful algorithms:

When a congestion control algorithm is stateful (e.g., TCP, with Slow Start, Congestion Avoidance and Fast Recovery), these states should be carefully considered such that the overall state of the aggregate flow is correct. This may require sharing more information in the UPDATE call.

Rate jumps:

The FSE-based coupling algorithms can let a flow quickly increase its rate to its fair share, e.g. when a new flow joins or after a quiescent period. In case of window-based congestion controls, this may produce a burst which should be mitigated in some way. An example of how this could be done without using a timer is presented in [anrw2016], using TCP as an example.

7. Expected feedback from experiments

The algorithm described in this memo has so far been evaluated using simulations covering all the tests for more than one flow from [I-D.ietf-rmcat-eval-test] (see [IETF-93], [IETF-94]). Experiments should confirm these results using at least the NADA congestion control algorithm with real-life code (e.g., browsers communicating over an emulated network covering the conditions in [I-D.ietf-rmcat-eval-test]). The tests with real-life code should be repeated afterwards in real network environments and monitored. Experiments should investigate cases where the media coder's output rate is below the rate that is calculated by the coupling algorithm (FSE_R(i) in algorithms 1 and 2, section 5.3). Implementers and testers are invited to document their findings in an Internet draft.

8. Acknowledgements

This document has benefitted from discussions with and feedback from Andreas Petlund, Anna Brunstrom, Colin Perkins, David Hayes, David Ros (who also gave the FSE its name), Ingemar Johansson, Karen Nielsen, Kristian Hiorth, Mirja Kuehlewind, Martin Stiernerling, Spencer Dawkins, Varun Singh, Xiaoqing Zhu, and Zaheduzzaman Sarker. The authors would like to especially thank Xiaoqing Zhu and Stefan Holmer for helping with NADA and GCC, and Anna Brunstrom as well as Julius Flohr for helping us correct the active algorithm for the case of application-limited flows.

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

9. IANA Considerations

This memo includes no request to IANA.

10. 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.

Implementers should also be aware of the Security Considerations sections of [RFC3124], [RFC5348], and [RFC7478].

11. References

11.1. Normative References

[I-D.ietf-rmcat-nada]

Zhu, X., *, R., Ramalho, M., Cruz, S., Jones, P., Fu, J., and S. D'Aronco, "NADA: A Unified Congestion Control Scheme for Real-Time Media", draft-ietf-rmcat-nada-11 (work in progress), July 2019.

[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>>.

- [RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, DOI 10.17487/RFC3124, June 2001, <<https://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, <<https://www.rfc-editor.org/info/rfc5348>>.

11.2. Informative References

- [anrw2016] Islam, S. and M. Welzl, "Start Me Up:Determining and Sharing TCP's Initial Congestion Window", ACM, IRTF, ISOC Applied Networking Research Workshop 2016 (ANRW 2016) , 2016.
- [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://safiquili.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-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-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-02 (work in progress), July 2016.
- [I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-19 (work in progress), November 2017.

- [I-D.ietf-rtcweb-transports]
Alvestrand, H., "Transports for WebRTC", Internet-draft draft-ietf-rtcweb-transports-17.txt, October 2016.
- [IETF-93] Islam, S., Welzl, M., and S. Gjessing, "Updates on Coupled Congestion Control for RTP Media", July 2015, <<https://www.ietf.org/proceedings/93/rmcat.html>>.
- [IETF-94] Islam, S., Welzl, M., and S. Gjessing, "Updates on Coupled Congestion Control for RTP Media", November 2015, <<https://www.ietf.org/proceedings/94/rmcat.html>>.
- [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>>.
- [RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <<https://www.rfc-editor.org/info/rfc7656>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<https://www.rfc-editor.org/info/rfc8087>>.
- [RFC8382] Hayes, D., Ed., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media", RFC 8382, DOI 10.17487/RFC8382, June 2018, <<https://www.rfc-editor.org/info/rfc8382>>.
- [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. Application to GCC

Google Congestion Control (GCC) [I-D.ietf-rmcat-gcc] is another congestion control scheme for RTP flows that is under development. GCC is not yet finalised, but at the time of this writing, 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).

Appendix B. Scheduling

When flows 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 C. 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 may (depending on the congestion control algorithm) 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 and not safe to deploy outside of testbed environments. Results of a simplified passive FSE algorithm with both NADA and GCC can be found in [fse-noms].

In the passive version of the FSE, TLO (the Total Leftover Rate) is a static variable per FG which is initialized to 0. Additionally, 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. $FSE_R(f)$ and $DR(f)$ 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 $FSE_R(f)$ to S_CR .
- (2) When a flow f stops or pauses, it sets its $DR(f)$ to 0 and sets $P(f)$ to -1.
- (3) Every time the congestion controller of the flow f determines a new sending rate $CC_R(f)$, assuming the flow's new desired rate $new_DR(f)$ 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, $Rate(f)$. 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 $DELTA$: the difference between $FSE_R(f)$ and $CC_R(f)$.


```

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

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


```

          FSE_R(f) = CC_R(f)
          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(f), 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(f).

```

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

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

```

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

```

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

```

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.

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

$CC_R(2) = 2$. $new_DR(2) = \text{infinity}$.
 3 a) $new_S_CR = 7$; $DELTA = 2 - 1 = 1$.
 3 b) $FSE_R(2) = 2$. $DELTA$ is positive, hence $S_CR = 9 + 1 = 10$;
 $DR(2) = 2$.
 3 c) $S_P = 1.5$.
 3 d) $Rate(2) = \min(\text{infinity}, 0.5/1.5 * 10 + 0) = 3.33$.
 3 e) $DR(2) = FSE_R(2) = 3.33$.

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.

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

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.

$CC_R(2) = 4.33$. $new_DR(2) = \text{infinity}$.

3 a) $new_S_CR = 5.33$; $\Delta = 1$.

3 b) $FSE_R(2) = 4.33$. Δ is positive, hence $S_CR = 12$;
 $DR(2) = 4.33$.

3 c) $S_P = 1.5$.

3 d) $Rate(2) = \min(\text{infinity}, 0.5/1.5 * 12 + 5.33) = 9.33$.

3 e) $FSE_R(2) = 9.33$, $DR(2) = 9.33$.

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 $P(1)$ to -1 and $DR(1)$ 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.

$CC_R(2) = 7.33$. $new_DR(2) = \text{infinity}$.
 3 a) $new_S_CR = 11.33$; $\Delta = -2$.
 3 b) $FSE_R(2) = 7.33$. Δ is negative, hence $S_CR = 9.33$;
 $DR(2) = 7.33$.
 3 c) Flow 1 has $P(1) = -1$, hence it is deleted from the FSE.
 $S_P = 0.5$.
 3 d) $Rate(2) = \min(\text{infinity}, 0.5/0.5 \cdot 9.33 + 0) = 9.33$.
 3 e) $FSE_R(2) = DR(2) = 9.33$.

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 D. Change log

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

D.1.1. Changes from -00 to -01

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

D.1.2. Changes from -01 to -02

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

D.1.3. Changes from -02 to -03

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

D.1.4. Changes from -03 to -04

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

D.1.5. Changes from -04 to -05

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

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

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

- o Moved scheduling section to the appendix.

D.2.2. Changes from -00 to -01

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

D.2.3. Changes from -01 to -02

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

D.2.4. Changes from -02 to -03

- o Minor changes.
- o Added a section about expected feedback from experiments.

D.2.5. Changes from -03 to -04

- o Described the names of variables used in the algorithms.
- o Added a diagram to illustrate the interaction between flows and the FSE.

- o Added text on the trade-off of using the configuration based approach.
- o Minor changes to enhance the readability.

D.2.6. Changes from -04 to -05

- o Changed several occurrences of "NADA and GCC" to "NADA", including the abstract.
- o Moved the application to GCC to an appendix, and made the GCC reference informative.
- o Provided a few more general recommendations on applying the coupling algorithm.

D.2.7. Changes from -05 to -06

- o Incorporated comments by Colin Perkins.

D.2.8. Changes from -06 to -07

- o Addressed OPSDIR, SECDIR, GENART, AD and IESG comments.

D.2.9. Changes from -07 to -08

- o Updated the algorithms in section 5 to support application-limited flows. Moved definition of Desired Rate from appendix to section 5. Updated references.

D.2.10. Changes from -08 to -09

- o Minor improvement of the algorithms in section 5.

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.

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

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Metrics	3
3.1. RTP Log Format	5
4. List of Network Parameters	5
4.1. One-way Propagation Delay	5
4.2. End-to-end Loss	5
4.3. DropTail Router Queue Length	6
4.4. Loss generation model	6
4.5. Jitter models	6
4.5.1. Random Bounded PDV (RBPDV)	7
4.5.2. Approximately Random Subject to No-Reordering Bounded PDV (NR-RPVD)	8
5. WiFi or Cellular Links	9
6. Traffic Models	9
6.1. TCP traffic model	9
6.2. RTP Video model	9
6.3. Background UDP	10
7. Security Considerations	10
8. IANA Considerations	10
9. Contributors	10
10. Acknowledgements	10
11. References	10
11.1. Normative References	10
11.2. Informative References	11
Appendix A. Application Trade-off	12
A.1. Measuring Quality	12
Appendix B. Change Log	13
B.1. Changes in draft-ietf-rmcat-eval-criteria-05	13
B.2. Changes in draft-ietf-rmcat-eval-criteria-04	13
B.3. Changes in draft-ietf-rmcat-eval-criteria-03	13
B.4. Changes in draft-ietf-rmcat-eval-criteria-02	13
B.5. Changes in draft-ietf-rmcat-eval-criteria-01	13
B.6. Changes in draft-ietf-rmcat-eval-criteria-00	13
B.7. Changes in draft-singh-rmcat-cc-eval-04	13
B.8. Changes in draft-singh-rmcat-cc-eval-03	14
B.9. Changes in draft-singh-rmcat-cc-eval-02	14
B.10. Changes in draft-singh-rmcat-cc-eval-01	14
Authors' Addresses	14

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:

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

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-Elliott 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?]

4.5.1.1. Random Bounded PDV (RBPVD)

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

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

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

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 $\text{serial}(n)$ be defined as the serialization delay of packet n at the lowest bottleneck link rate (or other appropriate rate) in a given test. Then we produce all the post-jitter values for $j(n)$ for $n = 1, 2, \dots, N$, where N is the length of the source sequence s to be 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 $[j(n-1)+\text{serial}(n-1)]$, we redefine $j(n)$ to be equal to $[j(n-1)+\text{serial}(n-1)]$ and continue for all remaining n (i.e., $n = 3, 4, \dots, N$). This models the case where the packet n is sent immediately after packet $(n-1)$ at the bottleneck link rate. Although this is generally the theoretical minimum in that it assumes that no other packets from other flows are in-between packet n and $n+1$ at the bottleneck link, it is a reasonable assumption for per flow queuing.

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

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

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

2 - Two-sided: Truncated Gaussian PDV Distribution. Four quantities to define: the appropriate `x_min` and `x_max` for test (e.g., +/- 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 traffic model

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

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

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

- o Removed the guidelines section, as most of the sections are now covered: wireless tests, video model, etc.
- o Improved Short TCP model based on the suggestion to use [httparchive.org](http://archive.org).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- o Added scenario descriptions.

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

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

Authors' Addresses

Varun Singh
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

RMCAT WG
Internet-Draft
Intended status: Informational
Expires: September 20, 2020

V. Singh
callstats.io
J. Ott
Technical University of Munich
S. Holmer
Google
March 19, 2020

Evaluating Congestion Control for Interactive Real-time Media
draft-ietf-rmcat-eval-criteria-14

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 20, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Metrics	4
3.1. RTP Log Format	5
4. List of Network Parameters	6
4.1. One-way Propagation Delay	6
4.2. End-to-end Loss	6
4.3. Drop Tail Router Queue Length	7
4.4. Loss generation model	7
4.5. Jitter models	7
4.5.1. Random Bounded PDV (RBPDV)	8
4.5.2. Approximately Random Subject to No-Reordering Bounded PDV (NR-RPVD)	9
4.5.3. Recommended distribution	10
5. Traffic Models	10
5.1. TCP traffic model	10
5.2. RTP Video model	11
5.3. Background UDP	11
6. Security Considerations	12
7. IANA Considerations	12
8. Contributors	12
9. Acknowledgments	12
10. References	13
10.1. Normative References	13
10.2. Informative References	14
Appendix A. Change Log	15
A.1. Changes in draft-ietf-rmcat-eval-criteria-07	15
A.2. Changes in draft-ietf-rmcat-eval-criteria-06	15
A.3. Changes in draft-ietf-rmcat-eval-criteria-05	15
A.4. Changes in draft-ietf-rmcat-eval-criteria-04	15
A.5. Changes in draft-ietf-rmcat-eval-criteria-03	15
A.6. Changes in draft-ietf-rmcat-eval-criteria-02	15
A.7. Changes in draft-ietf-rmcat-eval-criteria-01	16
A.8. Changes in draft-ietf-rmcat-eval-criteria-00	16
A.9. Changes in draft-singh-rmcat-cc-eval-04	16
A.10. Changes in draft-singh-rmcat-cc-eval-03	16
A.11. Changes in draft-singh-rmcat-cc-eval-02	16
A.12. Changes in draft-singh-rmcat-cc-eval-01	17
Authors' Addresses	17

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 congestion control algorithms are expected to operate within the envelope of the circuit breakers defined in RFC8083 [RFC8083].

This document only provides the broad set of network parameters and and traffic models for evaluating a new congestion control algorithm. The minimal requirements for congestion control proposals is to produce or present results for the test scenarios described in [I-D.ietf-rmcat-eval-test] (Basic Test Cases), which also defines the specifics for the test cases. Additionally, proponents may produce evaluation results for the wireless test scenarios [I-D.ietf-rmcat-wireless-tests].

This document does not cover application-specific implications of congestion control algorithms and how those could be evaluated. Therefore, no quality metrics are defined for performance evaluation; quality metrics and algorithms to infer those vary between media types. Metrics and algorithms to assess, e.g., quality of experience evolve continuously so that determining suitable choices is left for future work. However, there is consensus that each congestion control algorithm should be able to show that it is useful for interactive video by performing analysis using a real codecs and video sequences and state-of-the-art quality metrics.

Beyond optimizing individual metrics, real-time applications may have further options to trade off performance, e.g., across multiple media; refer to the RMCAT requirements [I-D.ietf-rmcat-cc-requirements] document. Such trade-offs may be defined in the future.

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

This document specifies testing criteria for evaluating congestion control algorithms for RTP media flows. Proposed algorithms are to prove their performance by means of simulation and/or emulation experiments for all the cases described.

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 [tcpdump], wireshark [wireshark]). The following metrics 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. Self-Fairness and Fairness with respect to cross traffic:
Experiments testing a given congestion control proposal must report on relative ratios of the average throughput (measured at coarser time intervals) obtained by each RTP media stream. In the presence of background cross-traffic such as TCP, the report must also include the relative ratio between average throughput of RTP media streams and cross-traffic streams.
During static periods of a test (i.e., when bottleneck bandwidth is constant and no arrival/departure of streams), these report on relative ratios serve as an indicator of how fair the RTP streams share bandwidth amongst themselves and against cross-traffic streams. The throughput measurement interval should be set at a few values (for example, at 1s, 5s, and 20s) in order to measure fairness across different time scales.
As a general guideline, the relative ratio between congestion controlled RTP flows with the same priority level and similar

path RTT should be bounded between (0.333 and 3.) 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 a congestion controlled RTP flow is by itself or competing with similar cross-traffic.

Note that the above metrics are all objective application-independent metrics. Refer to Section 3, in [I-D.ietf-netvc-testing] for objective metrics for evaluating codecs.

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

3.1. RTP Log Format

Having a common log format simplifies running analyses across and comparing different measurements. The log file should be tab or comma separated containing the following details:

Send or receive timestamp (Unix):	<int>.<int>	-- sec.usec decimal
RTP payload type	<int>	-- decimal
SSRC	<int>	-- hexadecimal
RTP sequence no	<int>	-- decimal
RTP timestamp	<int>	-- decimal
marker bit	0 1	-- character
Payload size	<int>	-- # bytes, decimal

Each line of the log file should be terminated with CRLF, CR, or LF characters. Empty lines are disregarded.

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).

These data should suffice to compute the media-encoding independent metrics described above. Use of a common log will allow simplified post-processing and analysis across different implementations.

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 across a broad range of path characteristics, also including challenging situations, for example over trans-continental and/or satellite links. Tests thus account for the following different latencies:

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

4.2. End-to-end Loss

Many paths in the Internet today are largely lossless but, with wireless networks and interference, towards remote regions, or in scenarios featuring high/fast mobility, media flows may exhibit substantial packet loss. This variety needs to be reflected appropriately by the tests.

To model a wide range of 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. Drop Tail Router Queue Length

Routers should be configured to use Drop Tail queues in the experiments due to their (still) prevalent nature. Experimentation with AQM schemes is encouraged but not mandatory.

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

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

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

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

4.4. Loss generation model

Many models for generating packet loss are available, some yield correlated, others independent losses; losses can also be extracted from packet traces. As a (simple) minimum loss model with minimal parameterization (i.e., the loss rate), independent random losses must be used in the evaluation.

It is known that independent loss models may reflect reality poorly and hence more sophisticated loss models could be considered. Suitable models for correlated losses includes the Gilbert-Elliott model [gilbert-elliott] and losses generated by modeling a queue including its (different) drop behaviors.

4.5. Jitter models

This section defines jitter models for the purposes of this document. When jitter is to be applied to both the congestion controlled RTP 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. It is typically used to refer to 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 computed as the smoothed difference in packet arrival times relative to their respective expected arrival times, which is particularly meaningful if the underlying packet delay variation was caused by a Gaussian random process.

Because jitter is an overloaded term, we use the term Packet Delay Variation (PDV) instead 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 results in one-sided histograms. In the usual packet network transport case, there is typically one packet that transited the network with the minimum delay; a (large) number of packets transit the network within some (smaller) positive variation from this minimum delay, and a (small) number of the packets transit 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 rate adaptation designs for RTP congestion control.

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

The former, 1) Random Bounded PDV is presented for information only, while the latter, 2) Approximately Random Subject to No-Reordering Bounded PDV, must be used in the evaluation.

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 defined to be two standard deviations higher than the mean.

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

We assume here that $s(n)$ is the original source time of packet n and the post-jitter induced emission time, $j(n)$, for packet n is:

$$j(n) = \{[z(n) + x_{\text{mean}}] + s(n)\}.$$

It follows that the separation in the post-jitter time of packets n and $n+1$ is $\{[s(n+1)-s(n)] - [z(n)-z(n+1)]\}$. Since the first term is always a positive quantity, we note that packet reordering at the receiver is possible whenever the second term is greater than the first. Said another way, whenever the difference in possible zero mean PDV sample delays (i.e., $[x_{\text{max}}-x_{\text{min}}]$) 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 $\text{serial}(n)$ be defined as the serialization delay of packet n at the lowest bottleneck link rate (or other appropriate rate) in a given test. Then we produce all the post-jitter values for $j(n)$ for $n = 1, 2, \dots, N$, where N is the length of the source sequence s to be 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 $[j(n-1)+\text{serial}(n-1)]$, we redefine $j(n)$ to be equal to $[j(n-1)+\text{serial}(n-1)]$ and continue for all remaining n (i.e., $n = 3, 4, \dots, N$). This models the case where the packet n is sent immediately after packet $(n-1)$ at the bottleneck link rate. Although this is generally the theoretical minimum in that it assumes that no other packets from other flows are in-between packet n and $n+1$ at the bottleneck link, it is a reasonable assumption for per flow queuing.

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

4.5.3. Recommended distribution

Whether Random Bounded PDV or Approximately Random Subject to No-Reordering Bounded PDV, it is recommended that $z(n)$ is distributed according to a truncated Gaussian for the above jitter models:

$$z(n) \sim |\max(\min(N(0, \text{std}^2), N_STD * \text{std}), -N_STD * \text{std})|$$

where $N(0, \text{std}^2)$ is the Gaussian distribution with zero mean and standard deviation std. Recommended values:

- o std = 5 ms
- o N_STD = 3

5. Traffic Models

5.1. TCP traffic model

Long-lived TCP flows will download data throughout the session and are expected to have infinite amount of data to send or receive. This roughly applies, for example, when downloading software distributions.

Each short TCP flow is modeled as a sequence of file downloads interleaved with idle periods. Not all short TCP flows 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 modeled as follows: 30 connections start simultaneously fetching small (30-50 KB) amounts of data, evenly distributed. This covers the case where the short TCP flows are fetching web page resources rather than video files.

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].

Many different TCP congestion control schemes are deployed today. Therefore, experimentation with a range of different schemes, especially including CUBIC, is encouraged. Experiments must document in detail which congestion control schemes they tested against and which parameters were used.

5.2. RTP Video model

[RFC8593] describes two types of video traffic models for evaluating candidate algorithms for RTP congestion control. The first model statistically characterizes the behavior of a video encoder, whereas the second model uses video traces.

Sample video test sequences are available at [xiph-seq]. The following two video streams are the recommended minimum for testing: Foreman (CIF sequence) and FourPeople (720p); both come as raw video data to be encoded dynamically. As these video sequences are short (300 and 600 frames, respectively, they shall be stitched together repeatedly until the desired length is reached.

5.3. Background UDP

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. The inter packet interval is calculated based on the CBR and the packet size (is typically set to the path MTU size, the default value can be 1500 bytes).

Note that new transport protocols such as QUIC may use UDP but, due to their congestion control algorithms, will exhibit behavior conceptually similar in nature to TCP flows above and can thus be subsumed by the above, including the division into short- and long-lived flows. As QUIC evolves independently of TCP congestion control algorithms, its future congestion control should be considered as competing traffic as appropriate.

6. Security Considerations

This document specifies evaluation criteria and parameters for assessing and comparing the performance of congestion control protocols and algorithms for real-time communication. This memo itself is thus not subject to security considerations but the protocols and algorithms evaluated may be. In particular, successful operation under all tests defined in this document may suffice for a comparative evaluation but must not be interpreted that the protocol is free of risks when deployed on the Internet as briefly described in the following by example.

Such evaluations are expected to be carried out in controlled environments for limited numbers of parallel flows. As such, these evaluations are by definition limited and will not be able to systematically consider possible interactions or very large groups of communicating nodes under all possible circumstances, so that careful protocol design is advised to avoid incidentally contributing traffic that could lead to unstable networks, e.g., (local) congestion collapse.

This specification focuses on assessing the regular operation of the protocols and algorithms under considerations. It does not suggest checks against malicious use of the protocols -- by the sender, the receiver, or intermediate parties, e.g., through faked, dropped, replicated, or modified congestion signals. It is up to the protocol specifications themselves to ensure that authenticity, integrity, and/or plausibility of received signals are checked and the appropriate actions (or non-actions) are taken.

7. IANA Considerations

There are no IANA impacts in this memo.

8. 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.

9. Acknowledgments

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,

Randell Jesup, Mirja Kuehlewind, Karen Nielsen, Piers O'Hanlon, Colin Perkins, Michael Ramalho, Zaheduzzaman Sarker, Timothy B. Terriberry, Michael Welzl, Mo Zanaty, and Xiaoqing Zhu 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.

10. References

10.1. 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.
- [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>>.
- [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, <<https://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, <<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>>.
- [RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <<https://www.rfc-editor.org/info/rfc8083>>.

[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>>.

10.2. Informative References

[gilbert-elliott]
Hasslinger, G. and O. Hohlfeld, "The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet", 14th GI/ITG Conference - Measurement, Modelling and Evaluation of Computer and Communication Systems , 3 2008.

[I-D.ietf-netvc-testing]
Daede, T., Norkin, A., and I. Brailovski, "Video Codec Testing and Quality Measurement", draft-ietf-netvc-testing-09 (work in progress), January 2020.

[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., Zhu, X., and J. Fu, "Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks", draft-ietf-rmcat-wireless-tests-11 (work in progress), March 2020.

[RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, DOI 10.17487/RFC5033, August 2007, <<https://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, <<https://www.rfc-editor.org/info/rfc5166>>.

[tcpdump] "Homepage of tcpdump and libpcap", <https://www.tcpdump.org/index.html> .

[wireshark]
"Homepage of Wireshark", <https://www.wireshark.org> .

[xiph-seq]
Daede, T., "Video Test Media Set", <https://media.xiph.org/video/derf/> .

Appendix A. Change Log

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

A.1. Changes in draft-ietf-rmcat-eval-criteria-07

Updated the draft according to the discussion at IETF-101.

- o Updated the discussion on fairness. Thanks to Xiaoqing Zhu for providing text.
- o Fixed a simple loss model and provided pointers to more sophisticated ones.
- o Fixed the choice of the jitter model.

A.2. Changes in draft-ietf-rmcat-eval-criteria-06

- o Updated Jitter.

A.3. Changes in draft-ietf-rmcat-eval-criteria-05

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

A.4. Changes in draft-ietf-rmcat-eval-criteria-04

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

A.5. Changes in draft-ietf-rmcat-eval-criteria-03

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

A.6. Changes in draft-ietf-rmcat-eval-criteria-02

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

A.7. Changes in draft-ietf-rmcat-eval-criteria-01

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

A.8. Changes in draft-ietf-rmcat-eval-criteria-00

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

A.9. Changes in draft-singh-rmcat-cc-eval-04

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

A.10. Changes in draft-singh-rmcat-cc-eval-03

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

A.11. Changes in draft-singh-rmcat-cc-eval-02

- o Added scenario descriptions.

A.12. Changes in draft-singh-rmcat-cc-eval-01

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

Authors' Addresses

Varun Singh
CALLSTATS I/O Oy
Runeberginkatu 4c A 4
Helsinki 00100
Finland

Email: varun.singh@iki.fi
URI: <https://www.callstats.io/about>

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.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Structure of Test cases	3
4. Recommended Evaluation Settings	7
4.1. Evaluation metrics	8
4.2. Path characteristics	8
4.3. Media source	9
5. Basic Test Cases	10
5.1. Variable Available Capacity with a Single Flow	10
5.2. Variable Available Capacity with Multiple Flows	13
5.3. Congested Feedback Link with Bi-directional Media Flows	14
5.4. Competing Media Flows with same Congestion Control Algorithm	17
5.5. Round Trip Time Fairness	19
5.6. Media Flow Competing with a Long TCP Flow	21
5.7. Media Flow Competing with Short TCP Flows	23
5.8. Media Pause and Resume	25
6. Other potential test cases	27
6.1. Media Flows with Priority	27
6.2. Explicit Congestion Notification Usage	27
6.3. Multiple Bottlenecks	27
7. Wireless Access Links	29
8. Security Considerations	30
9. IANA Considerations	30
10. Acknowledgements	30
11. References	30
11.1. Normative References	30
11.2. Informative References	31
Authors' Addresses	31

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.

o 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.

- ### o Define testbed topology: every test case needs to define an evaluation testbed topology. Figure 1 shows such an evaluation topology. In this evaluation topology, S1..Sn are traffic sources. These sources generate media traffic and use either 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.

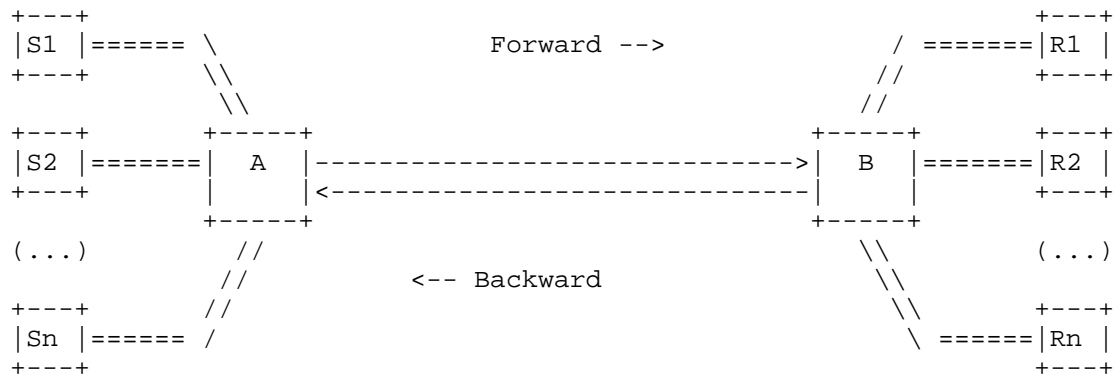


Figure 1: Example of A Testbed Topology

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.

o 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.

- o Path direction: forward and backward.
- o Reference bottleneck capacity: 1Mbps.
- o One-Way propagation delay: 50ms. Implementers are encouraged to run the experiment with additional propagation delays mentioned in [I-D.ietf-rmcat-eval-criteria]
- o Maximum end-to-end jitter: 30ms. Jitter models are described in [I-D.ietf-rmcat-eval-criteria]

- o Bottleneck queue type: Drop tail. Implementers are encouraged to run the experiment with other AQM schemes, such as FQ-CoDel and PIE.
- o Bottleneck queue size: 300ms.
- o Path loss ratio: 0%.

Examples of additional network parameters are discussed in [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.

- o 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:

- o an intermediate bottleneck
- o change in available capacity (e.g., due to interface change, routing change, abrupt arrival/departure of background non-adaptive traffic).
- o maximum media bit rate is greater than link capacity. In this case, the application will attempt to ramp up to its maximum bit rate, since the link capacity is limited to a value lower, the congestion control scheme is expected to stabilize the sending bit rate close to the available bottleneck capacity.

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.

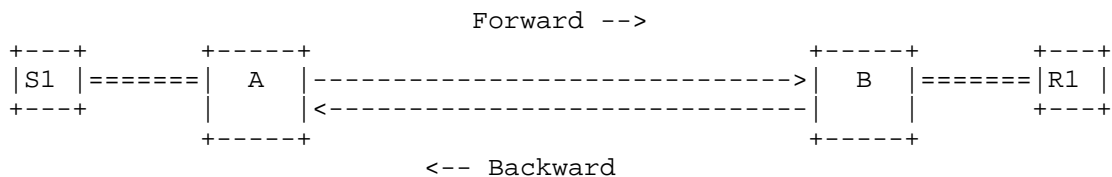


Figure 2: Testbed Topology for Limited Link Capacity

Testbed attributes:

- o Test duration: 100s
- o Path characteristics: as described in Section 4.2

- o Application-related:
 - * Media Traffic:
 - + Media type: Video
 - Media direction: forward.
 - Number of media sources: one (1)
 - Media timeline:
 - o Start time: 0s.
 - o End time: 99s.
 - + Media type: Audio
 - Media direction: forward.
 - Number of media sources: one (1)
 - Media timeline:
 - o Start time: 0s.
 - o End time: 99s.
 - * Competing traffic:
 - + Number of sources : zero (0)
- o Test Specific Information:
 - * 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.

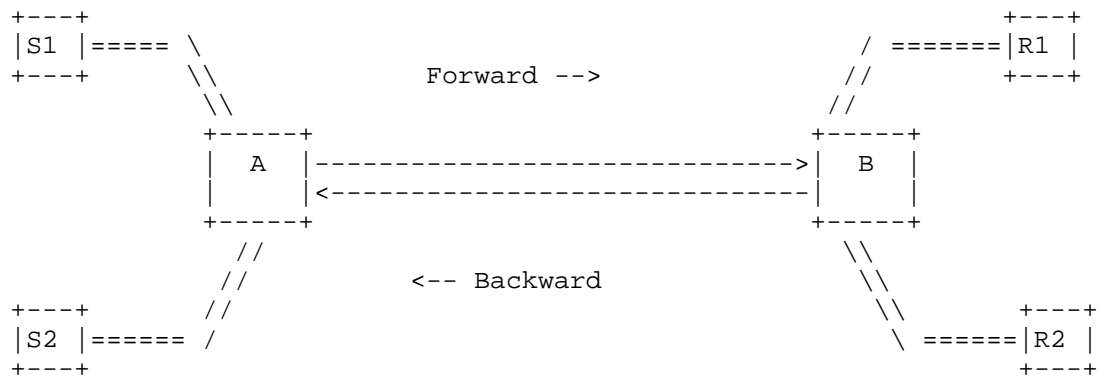


Figure 3: Testbed Topology for Variable Available Capacity

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.

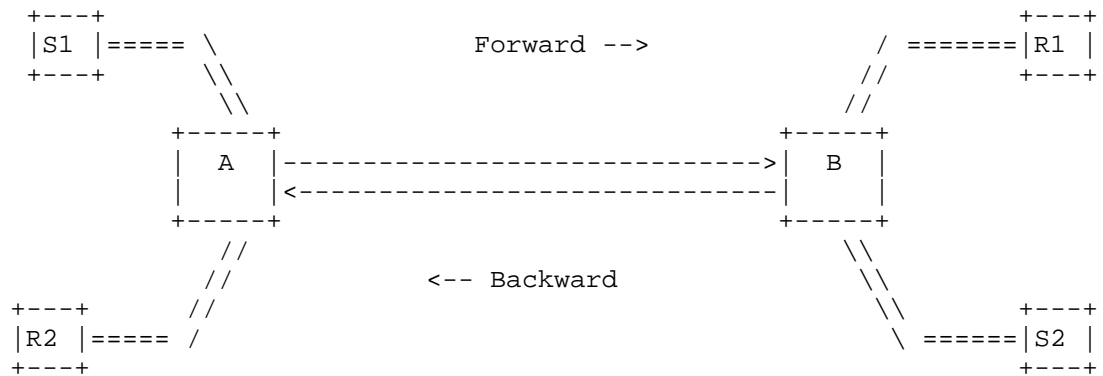


Figure 4: Testbed Topology for Congested Feedback Link

Testbed attributes:

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

- + Media type: Audio
 - Media direction: forward and backward
 - Number of media sources: two (2)
 - Media timeline:
 - o Start time: 0s.
 - o End time: 99s.
- * Competing traffic:
 - + Number of sources : zero (0)
- o Test Specific Information: this test uses path capacity variations to create congested feedback link. Table 3 lists the variation patterns applied to the forward path and Table 4 lists the variation patterns applied to the backward path. 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.

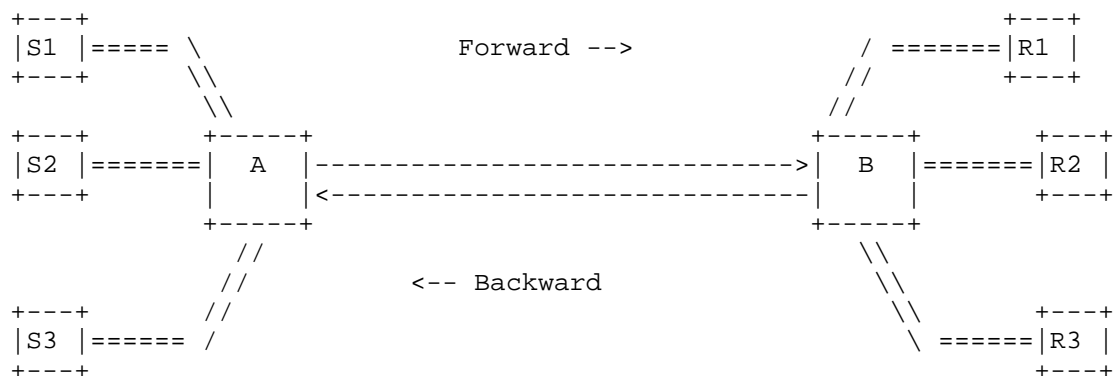


Figure 5: Testbed Topology for Multiple congestion controlled media Flows

Testbed attributes:

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

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

- * Media Source:
 - + Media type: Video
 - Media direction: forward
 - Number of media sources: five (5)
 - Media timeline: 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.

- Number of media sources: one (1)
- Media timeline:
 - o Start time: 5s.
 - o End time: 119s.
- * Additionally, implementers are encouraged to run the experiment with multiple media sources.
- * Competing traffic:
 - + Number and Types of sources : one (1) 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:

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

- + Number and Types of sources : ten (10), short-lived TCP flows.
 - + Traffic direction : forward
 - + Congestion algorithm: default TCP Congestion control [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 state. 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-

o Other test specific setup:

- * Media flow timeline:

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

- * Media flow timeline:

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

- * Media flow timeline:

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

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

6. Other potential test cases

It has been noticed that there are other interesting test cases besides the 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.

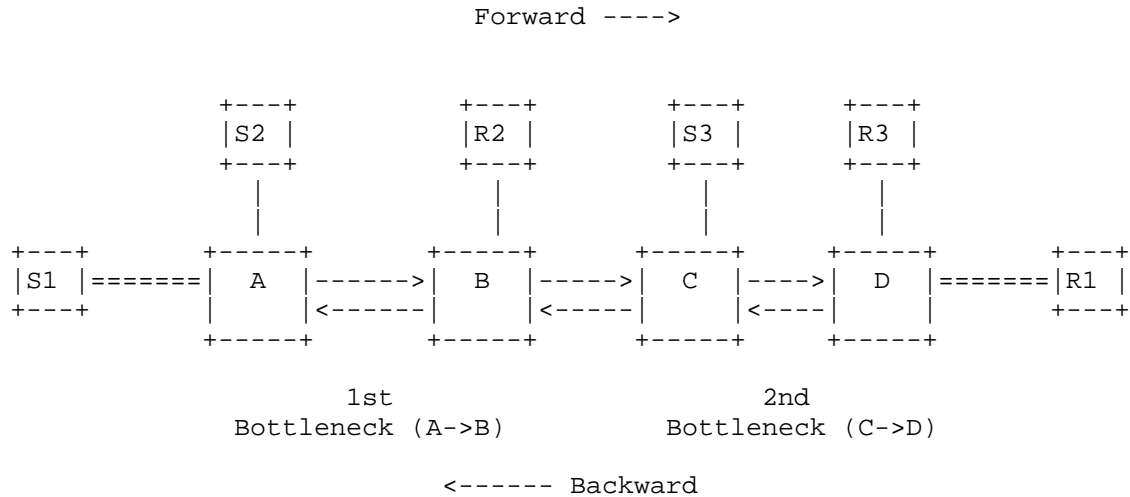


Figure 7: Testbed Topology for Multiple Bottlenecks

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:

- o Test duration: 300s
- o 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: November 24, 2019

Z. Sarker
Ericsson AB
V. Singh
callstats.io
X. Zhu
M. Ramalho
Cisco Systems
May 23, 2019

Test Cases for Evaluating RMCAT Proposals
draft-ietf-rmcat-eval-test-10

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 in a controlled environment.

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 November 24, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Structure of Test cases	3
4. Recommended Evaluation Settings	8
4.1. Evaluation metrics	8
4.2. Path characteristics	8
4.3. Media source	9
5. Basic Test Cases	10
5.1. Variable Available Capacity with a Single Flow	10
5.2. Variable Available Capacity with Multiple Flows	13
5.3. Congested Feedback Link with Bi-directional Media Flows	14
5.4. Competing Media Flows with same Congestion Control Algorithm	17
5.5. Round Trip Time Fairness	19
5.6. Media Flow Competing with a Long TCP Flow	21
5.7. Media Flow Competing with Short TCP Flows	23
5.8. Media Pause and Resume	25
6. Other potential test cases	27
6.1. Media Flows with Priority	27
6.2. Explicit Congestion Notification Usage	27
6.3. Multiple Bottlenecks	28
7. Wireless Access Links	30
8. Security Considerations	30
9. IANA Considerations	30
10. Acknowledgements	30
11. References	30
11.1. Normative References	30
11.2. Informative References	32
Authors' Addresses	32

1. Introduction

This memo describes a set of test cases for evaluating congestion control algorithm proposals in controlled environments 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 modelling 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.

o 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.

- o Define testbed topology: every test case needs to define an evaluation testbed topology. Figure 1 shows such an evaluation topology. In this evaluation topology, S1..Sn are traffic sources. These sources generate media traffic and use the congestion control algorithm(s) 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.

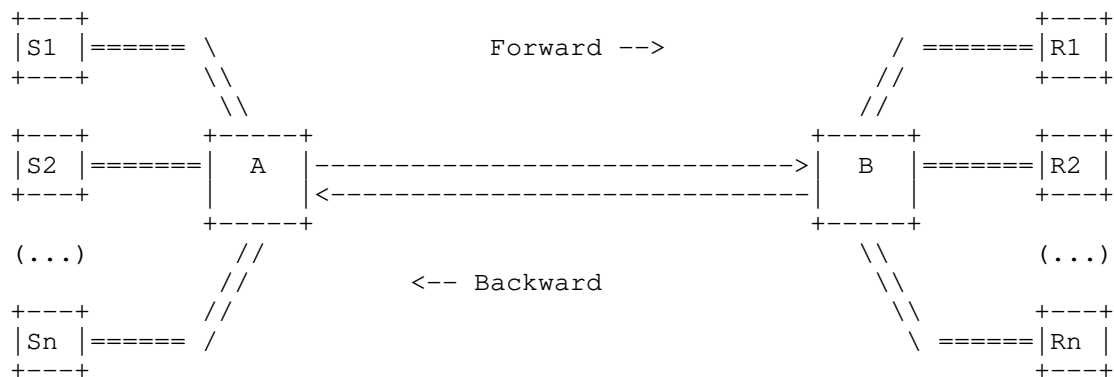


Figure 1: Example of A Testbed Topology

In a testbed environment with real equipments, there may exist a significant amount of unwanted traffic on the portions of the network path between the endpoints. 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. Such unwanted traffic should be removed or avoided to the greatest extent possible.

o 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.
- + Minimum 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, "tail drop" [RFC7567], Flow Queue -CoDel (FQ-CoDel) [RFC8290], or Proportional Integral controller Enhanced (PIE) [RFC8033].
- + 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. This 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 video scenario. For example, video test sequences are available at: [xiph-seq] and [HEVC-seq]. Different video scenarios give different distribution of video frames produced by the video encoder. Hence, it is important to specify the media content used in a particular test. If a synthetic video traffic source [I-D.ietf-rmcat-video-traffic-model] is used, then the synthetic video traffic source needs to be configured according to the characteristics of the media content specified.

- 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(s). For example - end-to-end delay observed on IP packet level, video frame level.
- B. Variation in sending bit rate and throughput. 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). Each occurrence of convergence during the test period need to be presented.

2. Transport level:

- A. Bandwidth utilization.
- B. Queue length (milliseconds at specified path capacity).

4.2. Path characteristics

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

- o Path direction: forward and backward.
- o Reference bottleneck capacity: 1Mbps.
- o One-Way propagation delay: 50ms. Implementers are encouraged to run the experiment with additional propagation delays mentioned in [I-D.ietf-rmcat-eval-criteria]

- o Maximum end-to-end jitter: 30ms. Jitter models are described in [I-D.ietf-rmcat-eval-criteria]
- o Bottleneck queue type: "tail drop". Implementers are encouraged to run the experiment with other AQM schemes, such as FQ-CoDel and PIE.
- o Bottleneck queue size: 300ms.
- o Path loss ratio: 0%.

Examples of additional network parameters are discussed in [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.

- o 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[xiph-seq].
- * 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 minimum bottleneck-link capacity between the two endpoints varies over time. This test is designed to measure the responsiveness of the candidate algorithm. This test tries to address the requirements in [I-D.ietf-rmcat-cc-requirements], which

requires the algorithm to adapt the flow(s) and provide lower end-to-end latency when there exists:

- o an intermediate bottleneck
- o change in available capacity (e.g., due to interface change, routing change, abrupt arrival/departure of background non-adaptive traffic).
- o maximum media bit rate is greater than link capacity. In this case, when the application tries 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, converge to the bottleneck link's capacity and adapt the flow to avoid unwanted media rate oscillation when the sending bit rate is approaching the bottleneck link's capacity. Such oscillations might 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.

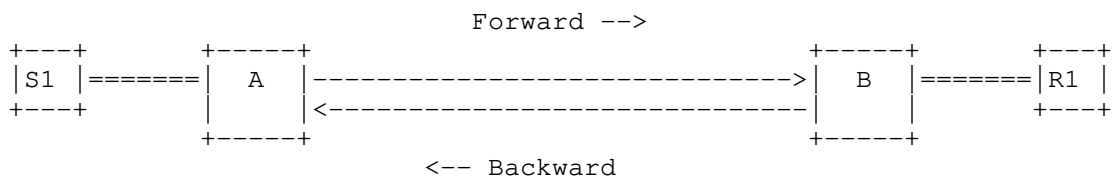


Figure 2: Testbed Topology for Limited Link Capacity

Testbed attributes:

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

$(Y \times r)$), where r is the Reference bottleneck capacity in Mbps and Y is the path capacity ratio 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.

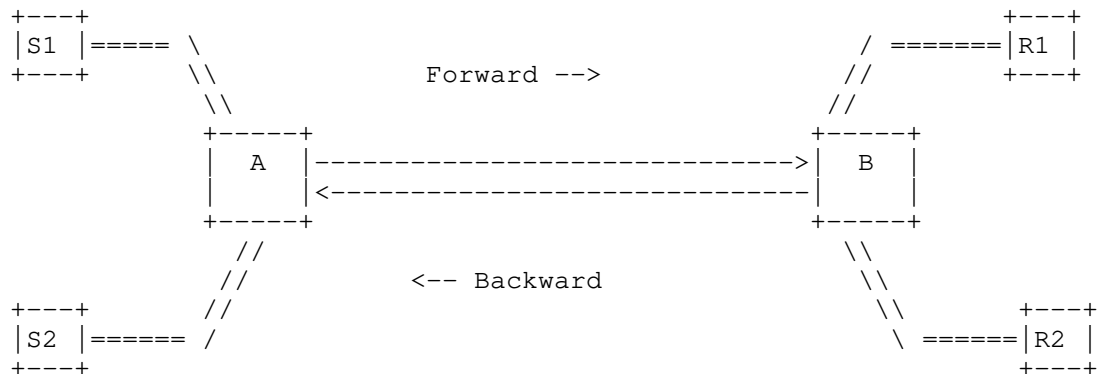


Figure 3: Testbed Topology for Variable Available Capacity

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 $(4 - (Y \times r))$, where r is the Reference bottleneck capacity in Mbps and Y is the path capacity ratio 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 the 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.

Expected behavior: 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.

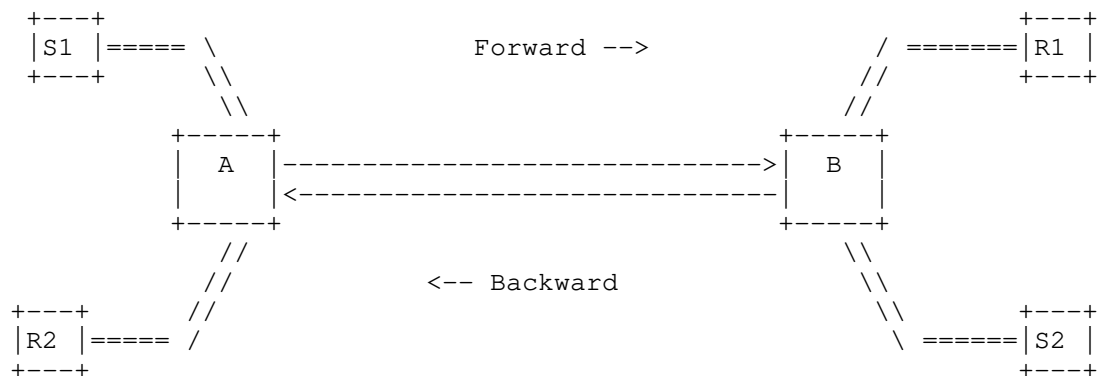


Figure 4: Testbed Topology for Congested Feedback Link

Testbed attributes:

- o Test duration: 100s
- o Path characteristics:

- * Reference bottleneck capacity: 1Mbps.
- o Application-related:
 - * Media Source:
 - + Media type: Video
 - Media direction: forward and backward
 - Number of media sources: two (2)
 - Media timeline:
 - o Start time: 0s.
 - o End time: 99s.
 - + Media type: Audio
 - Media direction: forward and backward
 - Number of media sources: two (2)
 - Media timeline:
 - o Start time: 0s.
 - o End time: 99s.
 - * Competing traffic:
 - + Number of sources : zero (0)
- o Test Specific Information: this test uses path capacity variations to create congested feedback link. Table 3 lists the variation patterns applied to the forward path and Table 4 lists the variation patterns applied to the backward path. 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 flow 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.

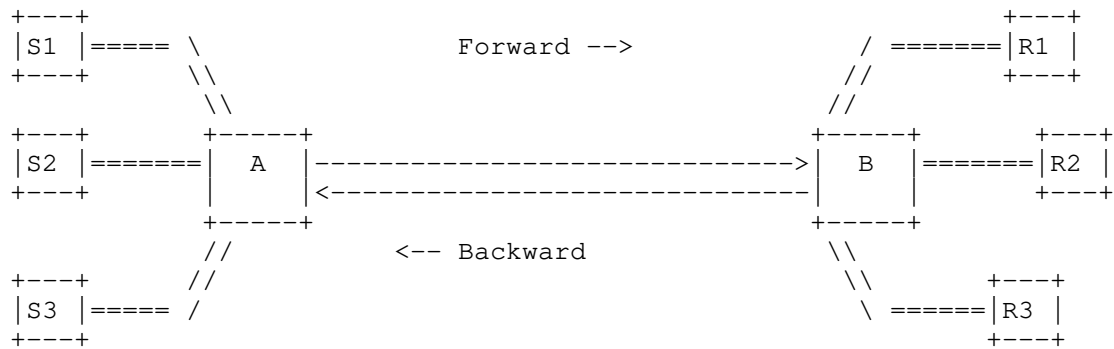


Figure 5: Testbed Topology for Multiple congestion controlled media Flows

Testbed attributes:

- o Test duration: 120s
- o Path characteristics:
 - * Reference bottleneck capacity: 3.5Mbps
 - * Path capacity ratio: 1.0
- o Application-related:
 - * Media Source:
 - + Media type: Video
 - Media direction: forward.
 - Number of media sources: three (3)
 - Media timeline: new media flows are added sequentially, at short time intervals. See test specific setup below.
 - + Media type: Audio

- Media direction: forward.
- Number of media sources: three (3)
- Media timeline: new media flows are added sequentially, at short time intervals. See test specific setup below.

* Competing traffic:

+ Number of sources : zero (0)

- o Test Specific Information: 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 one-way propagation delay 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.

Expected behavior: It is expected that the competing flows will converge to bit rates to accommodate all the flows with minimum possible latency and loss. The effectiveness of the algorithm depends on how fast and fairly the competing flows converge to their steady states irrespective of the RTT observed.

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.

Testbed attributes:

- o Test duration: 300s
- o Path characteristics:
 - * Reference bottleneck capacity: 4Mbps
 - * Path capacity ratio: 1.0
 - * One-Way propagation delay for each flow: 10ms for S1-R1, 25ms for S2-R2, 50ms for S3-R3, 100ms for S4-R4, and 150ms S5-R5.
- o 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.

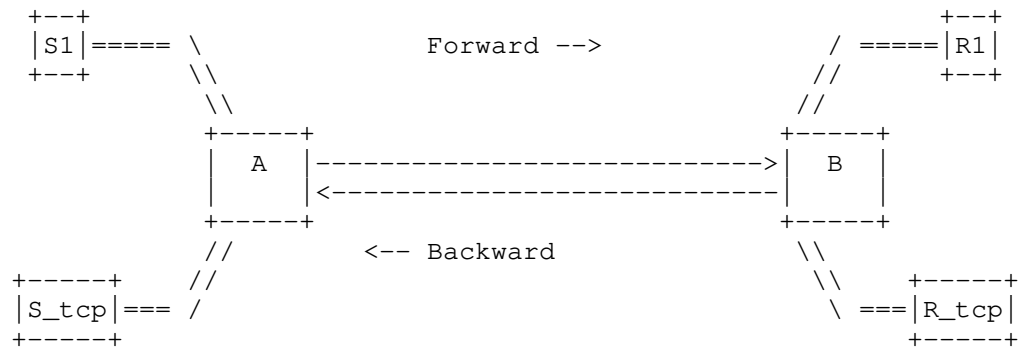


Figure 6: Testbed Topology for TCP vs congestion controlled media Flows

Testbed attributes:

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

- Media direction: forward
- Number of media sources: one (1)
- Media timeline:
 - o Start time: 5s.
 - o End time: 119s.
- * Additionally, implementers are encouraged to run the experiment with multiple media sources.
- * Competing traffic:
 - + Number and Types of sources : one (1) and long-lived TCP
 - + Traffic direction : forward
 - + Congestion control: default TCP congestion control[RFC5681]. Implementers are also encouraged to run the experiment with alternative TCP congestion control algorithm.
 - + 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 (for example, browsers) connect to a server and download a resource (typically a web page, few images, text files, etc.) using several TCP connections. 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.

Expected behavior: The candidate algorithm is expected to avoid flow starvation during the presence of short and bursty competing TCP flows, streaming at least at the minimum media bit rate. After competing TCP flows terminate, the media streams are expected to be robust enough to eventually recover to previous steady state behavior, and at the very least, avoid persistent starvation.

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:

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

- Number of media sources: two (2)
- Media timeline:
 - o Start time: 5s.
 - o End time: 299s.
- * Competing traffic:
 - + Number and Types of sources : ten (10), short-lived TCP flows.
 - + Traffic direction : forward
 - + Congestion algorithm: default TCP Congestion control [RFC5681]. Implementers are also encouraged to run the experiment with alternative TCP congestion control algorithm.
 - + Traffic timeline: each short TCP flow is modeled as a sequence of file downloads interleaved with idle periods. Not all short TCP flows start at the same time, 2 of them start in the ON state while rest of the 8 flows start in an OFF state. For description of short TCP flow model see test specific information below.
- o Test Specific Information:
 - * Short-TCP traffic model: The short TCP model to be used in this test is described in [I-D.ietf-rmcat-eval-criteria].

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 its 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.

Expected behavior: During the period where the third stream is paused, the two remaining flows are expected to increase their rates and reach the maximum media bit rate. When the third stream resumes, all three flows are expected to converge to the same original fair share of rates prior to the media pause/resume event.

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

1. Flow level:

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

Testbed Topology: Same as test case defined in Section 5.4

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

o Other test specific setup:

- * Media flow timeline:

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

- * Media flow timeline:

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

- + Resume time: at 60s
- * Media flow timeline:
 - + Flow ID: three (3)
 - + Start time: 0s
 - + Flow duration: 119s
 - + Pause time: not required
 - + Resume time: not required

6. Other potential test cases

It has been noticed that there are other interesting test cases besides the 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 this case, 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 [I-D.ietf-rmcat-coupled-cc] or use a weighted priority scheduler for the bandwidth distribution according to the respective media flow priority or use.

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->R1, 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 and ensuring fair share of the available bandwidth at each bottlenecks.

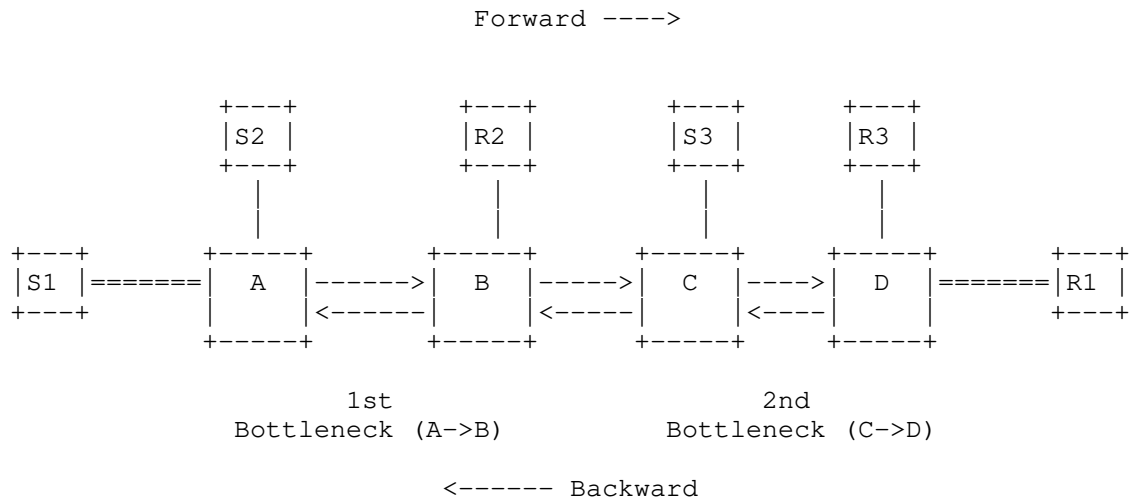


Figure 7: Testbed Topology for Multiple Bottlenecks

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

Testbed attributes:

- o Test duration: 300s
- o 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

The security considerations in [I-D.ietf-rmcat-eval-criteria] and the relevant congestion control algorithms apply. The principles for congestion control are described in [RFC2914], and in particular any new method must implement safeguards to avoid congestion collapse of the Internet.

The evaluation of the test cases are intended to be run in a controlled lab environment. Hence, the applications, simulators and network nodes ought to be well-behaved and should not impact the desired results. Moreover, proper measures must be taken to avoid leaking non-responsive traffic from unproven congestion avoidance techniques onto the open Internet.

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

[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-criteria]
Singh, V., Ott, J., and S. Holmer, "Evaluating Congestion Control for Interactive Real-time Media", draft-ietf-rmcat-eval-criteria-08 (work in progress), November 2018.
- [I-D.ietf-rmcat-video-traffic-model]
Zhu, X., Cruz, S., and Z. Sarker, "Video Traffic Models for RTP Congestion Control Evaluations", draft-ietf-rmcat-video-traffic-model-07 (work in progress), February 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-06 (work in progress), December 2018.
- [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>>.
- [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, <<https://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, <<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>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<https://www.rfc-editor.org/info/rfc5681>>.

- [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>>.

11.2. Informative References

- [HEVC-seq] HEVC, "Test Sequences",
http://www.netlab.tkk.fi/~varun/test_sequences/ .
- [I-D.ietf-rmcat-coupled-cc] Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-08 (work in progress), January 2019.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.
- [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>>.
- [xiph-seq] Xiph.org, "Video Test Media",
<http://media.xiph.org/video/derf/> .

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Torshamnsgatan 23
Stockholm, SE 164 83
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: Experimental
Expires: September 19, 2016

X. Zhu
R. Pan
M. Ramalho
S. Mena
P. Jones
J. Fu
Cisco Systems
S. D'Aronco
EPFL
C. Ganzhorn
March 18, 2016

NADA: A Unified Congestion Control Scheme for Real-Time Media
draft-ietf-rmcat-nada-02

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 <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

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	3
3. System Overview	3
4. Core Congestion Control Algorithm	5
4.1. Mathematical Notations	5
4.2. Receiver-Side Algorithm	8
4.3. Sender-Side Algorithm	10
5. Practical Implementation of NADA	12
5.1. Receiver-Side Operation	12
5.1.1. Estimation of one-way delay and queuing delay	12
5.1.2. Estimation of packet loss/marketing ratio	12
5.1.3. Estimation of receiving rate	13
5.2. Sender-Side Operation	13
5.2.1. Rate shaping buffer	14
5.2.2. Adjusting video target rate and sending rate	15
5.3. Feedback Message Requirements	15
6. Discussions and Further Investigations	16
6.1. Choice of delay metrics	16
6.2. Method for delay, loss, and marking ratio estimation	16
6.3. Impact of parameter values	17
6.4. Sender-based vs. receiver-based calculation	18
6.5. Incremental deployment	18
7. Implementation Status	18
8. IANA Considerations	19
9. Acknowledgements	19
10. References	19
10.1. Normative References	19
10.2. Informative References	20
Appendix A. Network Node Operations	21
A.1. Default behavior of drop tail queues	22
A.2. RED-based ECN marking	22

A.3. Random Early Marking with Virtual Queues	22
Authors' Addresses	23

1. Introduction

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

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

2. Terminology

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

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.

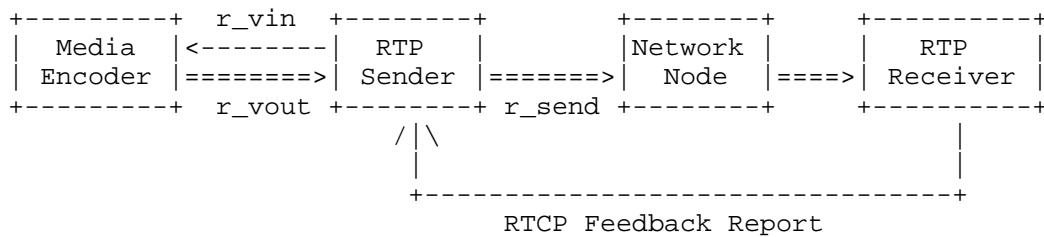


Figure 1: System Overview

- o Media encoder with rate control capabilities. It encodes raw media (audio and video) frames into compressed bitstream which is later packetized into RTP packets. As discussed in [I-D.ietf-rmcat-video-traffic-model], the actual output rate from the encoder r_{vout} may fluctuate around the target r_{vin} . Furthermore, it is possible that the encoder can only react to bit rate changes at rather coarse time intervals, e.g., once every 0.5 seconds.
- o 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.
- o 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} .
- o Network node with several modes of operation. The system can work with the default behavior of a simple drop tail queue. It can also benefit from advanced AQM features such as PIE, FQ-CoDel, RED-based ECN marking, and PCN marking using a token bucket algorithm. Note that network node operation is out of control for the design of NADA.

4. Core Congestion Control Algorithm

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

- o 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 (γ) is calculated based on observed round-trip-time and target feedback interval, so as to limit self-inflicted queuing delay.
- o 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.

Notation	Variable Name
t_curr	Current timestamp
t_last	Last time sending/receiving a feedback
delta	Observed interval between current and previous feedback reports: $\text{delta} = t_{\text{curr}} - t_{\text{last}}$
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 queueing 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: $x_{\text{diff}} = x_{\text{curr}} - x_{\text{prev}}$
rmode	Rate update mode: (0 = accelerated ramp-up; 1 = gradual update)
gamma	Rate increase multiplier in accelerated ramp-up mode
rtt	Estimated round-trip-time at sender
buffer_len	Rate shaping buffer occupancy measured in bytes

Figure 2: List of variables.

Notation	Parameter Name	Default Value
PRIO	Weight of priority of the flow	1.0
RMIN	Minimum rate of application supported by media encoder	150 Kbps
RMAX	Maximum rate of application supported by media encoder	1.5 Mbps
XREF	Reference congestion level	20ms
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
DFILT	Bound on filtering delay	120ms
LOGWIN	Observation window in time for calculating packet summary statistics at receiver	500ms
QEPS	Threshold for determining queuing delay build up at receiver	10ms
QTH	Delay threshold for non-linear warping	50ms
QMAX	Delay upper bound for non-linear warping	400ms
DLOSS	Delay penalty for loss	1.0s
DMARK	Delay penalty for ECN marking	200ms
GAMMA_MAX	Upper bound on rate increase ratio for accelerated ramp-up	50%
QBOUND	Upper bound on self-inflicted queuing delay during ramp up	50ms
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.

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

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 RTCP feedback report 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, a moderate queuing delay value below 100ms is likely self-inflicted or induced by other delay-based flows, whereas a high queuing delay value of several hundreds of milliseconds may indicate the presence of a loss-based flow that does not refrain from increased delay.

When packet losses are observed, the estimated queuing delay follows a non-linear warping inspired by the delay-adaptive congestion window backoff policy in [Budzisz-TON11]:

$$d_tilde = \begin{cases} d_queue, & \text{if } d_queue < QTH; \\ QTH \frac{(QMAX - d_queue)^4}{(QMAX - QTH)^4}, & \text{if } QTH < d_queue < QMAX \quad (1) \\ 0, & \text{otherwise.} \end{cases}$$

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

The aggregate congestion signal is:

$$x_curr = d_tilde + p_mark * DMARK + p_loss * DLOSS. \quad (2)$$

Here, $DMARK$ is prescribed delay penalty associated with ECN markings and $DLOSS$ is prescribed delay penalty associated with packet losses. 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 for them to 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:

- o No recent packet losses within the observation window $LOGWIN$; and
- o 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 [RMIN, RMAX]
  x_prev = x_curr
  t_last = t_curr

```

In accelerated ramp-up mode, the rate r_{ref} is updated as follows:

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

$$r_{ref} = \max(r_{ref}, (1 + \gamma) r_{recv}) \quad (4)$$

The rate increase multiplier γ 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:

$$x_offset = x_curr - PRIO * XREF * RMAX / r_ref \quad (5)$$

$$x_diff = x_curr - x_prev \quad (6)$$

$$r_ref = r_ref - KAPPA * \frac{\Delta}{TAU} * \frac{x_offset}{TAU} * r_ref - KAPPA * \frac{x_diff}{TAU} * r_ref \quad (7)$$

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 maximum rate ($RMAX$). Values of $RMIN$ and $RMAX$ will be provided by the media codec, as specified in [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 2Mbps for $RMAX$.

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

$$r_ref = \min(r_ref, RMAX) \quad (8)$$

$$r_ref = \max(r_ref, RMIN) \quad (9)$$

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

5. Practical Implementation of NADA

5.1. Receiver-Side Operation

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

5.1.1. Estimation of one-way delay and queuing delay

The delay estimation process in NADA follows a similar approach as in earlier delay-based congestion control schemes, such as LEDBAT [RFC6817]. Instead of relying on RTP timestamps, the NADA sender generates its own timestamp based on local system clock and embeds that information in the transport packet header. The NADA receiver estimates the forward delay as having a constant base delay component plus a time varying queuing delay component. The base delay is estimated as the minimum value of one-way delay observed over a relatively long period (e.g., tens of minutes), whereas the individual queuing delay value is taken to be the difference between one-way delay and base delay. All delay estimations are based on sender timestamps with higher granularity than RTP timestamps.

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. Current implementation employs a 15-tap minimum filter over per-packet queuing delay estimates.

5.1.2. Estimation of packet loss/marketing ratio

The receiver detects packet losses via gaps in the RTP sequence numbers of received packets. Packets arriving out-of-order are discarded, and count towards 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:

$$p_{loss} = \text{ALPHA} * p_{inst} + (1 - \text{ALPHA}) * p_{loss}. \quad (10)$$

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) is included as part of the feedback report.

5.2. Sender-Side Operation

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

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

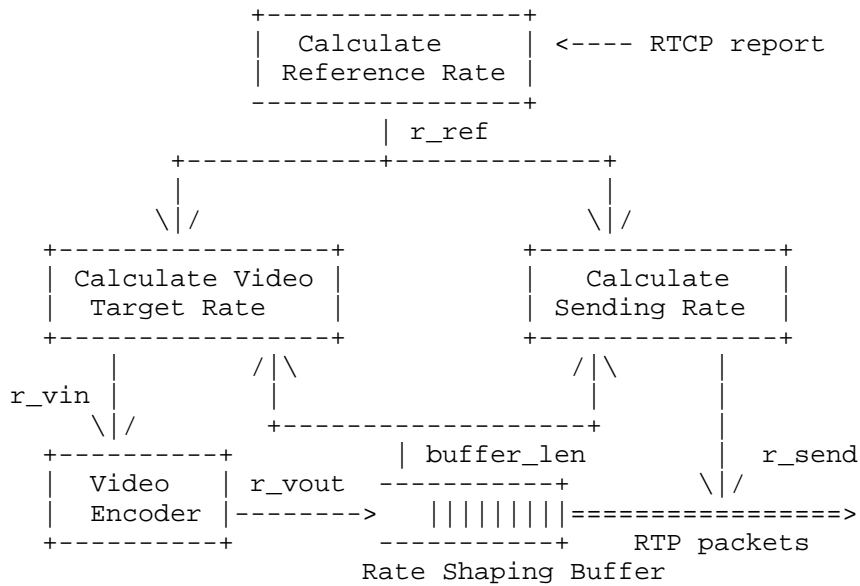


Figure 4: NADA Sender Structure

5.2.1. Rate shaping buffer

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

A rate shaping buffer is employed to absorb any instantaneous mismatch between encoder rate output r_{vout} and regulated sending rate r_{send} . Its current level of occupancy is measured in bytes and is denoted as $buffer_len$.

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

- o To deplete the rate shaping buffer faster by increasing the sending rate r_{send} ; and
- o To limit incoming packets of the rate shaping buffer by reducing the video encoder target rate r_{vin} .

5.2.2. Adjusting video target rate and sending rate

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

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

The actual sending rate r_{send} is regulated in a similar fashion:

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

In (11) and (12), the first term indicates the rate calculated from network congestion feedback alone. The second term indicates the influence of the rate shaping buffer. A large rate shaping buffer nudges the encoder target rate slightly below -- and the sending rate slightly above -- the reference rate r_{ref} .

Intuitively, the amount of extra rate offset needed to completely drain the rate shaping buffer within the duration of a single video frame is given by $8 * buffer_len * FPS$, where FPS stands for the frame rate of the video. The scaling parameters $BETA_V$ and $BETA_S$ can be tuned to balance between the competing goals of maintaining a small rate shaping buffer and deviating from the reference rate point.

5.3. Feedback Message Requirements

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

- o 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 ($rmode=1$).
- o Aggregated congestion signal (x_{curr}): the most recently updated value, calculated by the receiver according to Section 4.2. This information is expressed with a unit of 100 microsecond (i.e., 1/10 of a millisecond) in 15 bits. This allows a maximum value of x_{curr} at approximately 3.27 second.
- o Receiving rate (r_{recv}): the most recently measured receiving rate according to Section 5.1.3. This information is expressed with a unit of 10 Kilobits per second (Kbps) in 16 bits. This allows a maximum rate of approximately 6.55Mbps.

The above list of information can be accommodated by 32 bits in total. Choice of the feedback message interval Δ is discussed in Section 6.3 A target feedback interval of $\Delta=100ms$ is recommended.

6. Discussions and Further Investigations

6.1. Choice of delay metrics

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

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

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

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

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

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

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

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

6.3. Impact of parameter values

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

The choice of the target feedback interval DELTA needs to strike the right balance between timely feedback and low RTCP feedback message counts. A target feedback interval of DELTA=100ms is recommended, corresponding to a feedback bandwidth of 16Kbps with 200 bytes per feedback message --- less than 0.1% overhead for a 1 Mbps flow. Furthermore, both simulation studies and frequency-domain analysis have established that a feedback interval below 250ms 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 and QMAX. It is possible to adapt the value of both based on past observations of queuing delay in the presence of packet losses.

In calculating the aggregate congestion signal x_{curr} , the choice of DMARK and DLOSS influence the steady-state packet loss/marketing ratio experienced by the flow at a given available bandwidth. Higher values of DMARK and DLOSS result in lower steady-state loss/marketing ratios, but are more susceptible to the impact of individual packet loss/marketing events. While the value of DMARK and DLOSS are fixed and predetermined in the current design, a scheme for automatically tuning these values based on desired bandwidth sharing behavior in the presence of other competing loss-based flows (e.g., loss-based TCP) is under investigation.

[Editor's note: Choice of start value: is this in scope of congestion control, or should this be decided by the application?]

6.4. Sender-based vs. receiver-based calculation

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

6.5. Incremental deployment

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

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

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

Ultimately, networks equipped with proactive marking based on token bucket level metering can reap the additional benefits of zero standing queues and lower end-to-end delay and work seamlessly with existing senders and receivers.

7. Implementation Status

The NADA scheme has been implemented in [ns-2] and [ns-3] simulation platforms. Extensive ns-2 simulation evaluations of an earlier version of the draft are documented in [Zhu-PV13]. Evaluation results of the current draft over several test cases in [I-D.ietf-rmcat-eval-test] have been presented at recent IETF meetings [IETF-90][IETF-91]. Evaluation results of the current draft over several test cases in [I-D.ietf-rmcat-wireless-tests] have been presented at [IETF-93].

The scheme has also been implemented and evaluated in a lab setting as described in [IETF-90]. Preliminary evaluation results of NADA in single-flow and multi-flow scenarios have been presented in [IETF-91].

8. IANA Considerations

This document makes no request of IANA.

9. Acknowledgements

The authors would like to thank Randell Jesup, Luca De Cicco, Piers O'Hanlon, Ingemar Johansson, Stefan Holmer, Cesar Ilharco Magalhaes, Safiqul Islam, Mirja Kuhlewind, and Karen Elisabeth Egede Nielsen for their valuable questions and comments on earlier versions of this draft.

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>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc3550>>.
- [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>>.
- [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-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-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.
- [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-01 (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.

10.2. Informative References

- [RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, DOI 10.17487/RFC2309, April 1998, <<http://www.rfc-editor.org/info/rfc2309>>.
- [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>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc6817>>.

- [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.
- [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.
- [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.
- [ns-2] "The Network Simulator - ns-2",
<<http://www.isi.edu/nsnam/ns/>>.
- [ns-3] "The Network Simulator - ns-3", <<https://www.nsnam.org/>>.
- [IETF-90] Zhu, X., Ramalho, M., Ganzhorn, C., Jones, P., and R. Pan,
"NADA Update: Algorithm, Implementation, and Test Case
Evaluation 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](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](https://www.ietf.org/proceedings/93/slides/slides-93-rmcat-0.pdf)>.

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.

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 [RFC2309]. 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  $q < q_{lo}$ ;
      |
      |  $q_{avg} - q_{lo}$ 
      |  $p_{max} * \frac{q_{avg} - q_{lo}}{q_{hi} - q_{lo}}$ , if  $q_{lo} \leq q < q_{hi}$ ;
      |
      \  $p = 1$ ,                            if  $q \geq q_{hi}$ .
  
```

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

- * upon packet arrival, meter packet against token bucket (r,b);
- * update token level b_tk;
- * calculate the marking probability as:

$$p = \begin{cases} 0, & \text{if } b-b_{tk} < b_{lo}; \\ p_{max} * \frac{b-b_{tk}-b_{lo}}{b_{hi}-b_{lo}}, & \text{if } b_{lo} \leq b-b_{tk} < b_{hi}; \\ 1, & \text{if } b-b_{tk} \geq b_{hi}. \end{cases}$$

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
Cisco Systems
3625 Cisco Way
San Jose, CA 95134
USA

Email: ropan@cisco.com

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

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

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

Email: semena@cisco.com

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

Email: paulej@packetizer.com

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

Email: jianfu@cisco.com

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

Email: stefano.daronco@epfl.ch

Charles Ganzhorn
7900 International Drive, International Plaza, Suite 400
Bloomington, MN 55425
USA

Email: charles.ganzhorn@gmail.com

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.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. System Overview	3
4. Core Congestion Control Algorithm	5
4.1. Mathematical Notations	5
4.2. Receiver-Side Algorithm	8
4.3. Sender-Side Algorithm	10
5. Practical Implementation of NADA	13
5.1. Receiver-Side Operation	13
5.1.1. Estimation of one-way delay and queuing delay	13
5.1.2. Estimation of packet loss/marketing ratio	13
5.1.3. Estimation of receiving rate	14
5.2. Sender-Side Operation	14
5.2.1. Rate shaping buffer	15
5.2.2. Adjusting video target rate and sending rate	16
5.3. Feedback Message Requirements	16
6. Discussions and Further Investigations	17
6.1. Choice of delay metrics	17
6.2. Method for delay, loss, and marketing ratio estimation	18
6.3. Impact of parameter values	18
6.4. Sender-based vs. receiver-based calculation	20
6.5. Incremental deployment	20
7. Reference Implementations	20
8. Suggested Experiments	21
9. IANA Considerations	22
10. Security Considerations	22
11. Acknowledgments	22
12. Contributors	22
13. References	23
13.1. Normative References	23
13.2. Informative References	24
Appendix A. Network Node Operations	26
A.1. Default behavior of drop tail queues	27
A.2. RED-based ECN marking	27
A.3. Random Early Marking with Virtual Queues	28
Authors' Addresses	28

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.

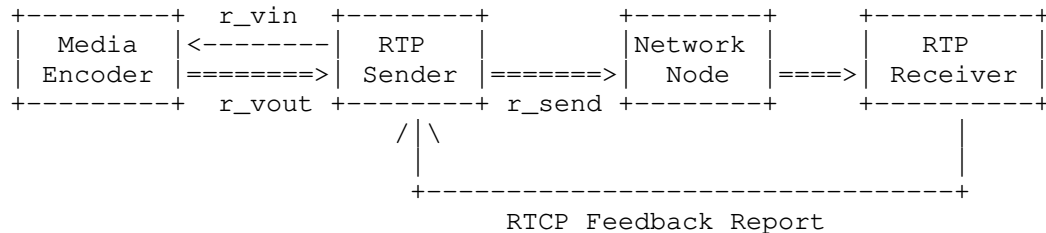


Figure 1: System Overview

- o 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.
- o 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.
- o 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_rcv*) 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_rcv*.
- o 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:

- o 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 (γ) is calculated based on observed round-trip-time and target feedback interval, so as to limit self-inflicted queuing delay.
- o 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: $\text{delta} = \text{t_curr} - \text{t_last}$
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: $\text{x_diff} = \text{x_curr} - \text{x_prev}$
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:

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

In (1), the queuing delay value is unchanged when it is below the first threshold QTH ; otherwise it is scaled down following a non-linear curve. This non-linear warping is inspired by the delay-adaptive congestion 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:

$$x_curr = d_tilde + DMARK * \frac{p_mark^2}{PMRREF} + DLOSS * \frac{p_loss^2}{PLRREF}. \quad (2)$$

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:

- o No recent packet losses within the observation window LOGWIN; and
- o 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:

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

$$r_ref = \max(r_ref, (1 + \gamma) r_recv) \quad (4)$$

The rate increase multiplier γ 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:

$$x_offset = x_curr - \text{PRIO} * \text{XREF} * \text{RMAX} / r_ref \quad (5)$$

$$x_diff = x_curr - x_prev \quad (6)$$

$$r_ref = r_ref - \text{KAPPA} * \frac{\text{delta}}{\text{TAU}} * \frac{x_offset}{\text{TAU}} * r_ref - \text{KAPPA} * \text{ETA} * \frac{x_diff}{\text{TAU}} * r_ref \quad (7)$$

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 $\text{PRIO} * \text{XREF}$.

At equilibrium, the aggregated congestion signal stabilizes at $x_curr = \text{PRIO} * \text{XREF} * \text{RMAX} / r_ref$. 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:

$$r_ref = \min(r_ref, \text{RMAX}) \quad (8)$$

$$r_ref = \max(r_ref, \text{RMIN}) \quad (9)$$

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 $d_{\text{queue}} = d_{\text{fwd}} - d_{\text{base}}$, 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/marketing 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:

$$p_{\text{loss}} = \text{ALPHA} * p_{\text{inst}} + (1 - \text{ALPHA}) * p_{\text{loss}}. \quad (10)$$

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.

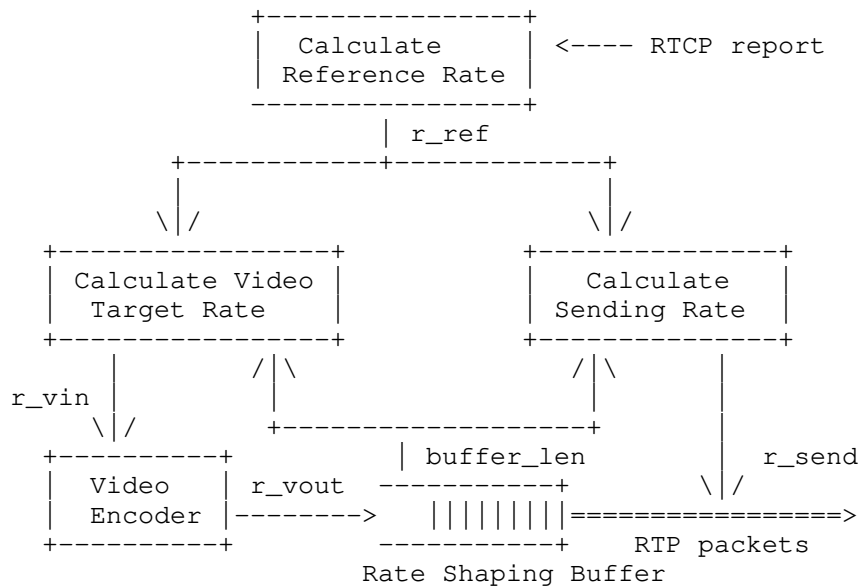


Figure 4: NADA Sender Structure

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:

- o To deplete the rate shaping buffer faster by increasing the sending rate r_{send} ; and
- o 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:

$$r_{diff_v} = \min(0.05 \cdot r_{ref}, BETA_V \cdot 8 \cdot buffer_len \cdot FPS). \quad (11)$$

$$r_{diff_s} = \min(0.05 \cdot r_{ref}, BETA_S \cdot 8 \cdot buffer_len \cdot FPS). \quad (12)$$

$$r_{vin} = \max(RMIN, r_{ref} - r_{diff_v}). \quad (13)$$

$$r_{send} = \min(RMAX, r_{ref} + r_{diff_s}). \quad (14)$$

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 $8 \cdot buffer_len \cdot FPS$, 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:

- o 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).
- o 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.
- o 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 $\text{TAU}=500\text{ms}$, $\text{DELTA}=100\text{ms}$ and $\text{ETA} = 2.0$ 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 $\text{DELTA}=100\text{ms}$ 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/marketing ratio experienced by the flow at a given available bandwidth. Higher values of DMARK and DLOSS result in lower steady-state loss/marketing ratios, but are more susceptible to the impact of individual packet loss/marketing 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:

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

- o Experiments with multiple NADA streams bearing different user-specified priorities.
- o Experiments with additional access technologies, especially over cellular networks such as 3G/LTE.
- o 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 Stiernerling 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 Evaluation 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., McKeeney, 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:

$$q_{avg} = w \cdot q + (1-w) \cdot q_{avg}.$$

calculate marking probability p as:

$$p = \begin{cases} 0, & \text{if } q < q_{lo}; \\ p_{max} \cdot \frac{q_{avg} - q_{lo}}{q_{hi} - q_{lo}}, & \text{if } q_{lo} \leq q < q_{hi}; \\ 1, & \text{if } q \geq q_{hi}. \end{cases}$$

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

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

A.3. Random Early Marking with Virtual Queues

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

Calculation of the marking probability involves the following steps:

```

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

    update token level b_tk;

    calculate the marking probability as:

        / 0,                                if b-b_tk < b_lo;
        |
        | b-b_tk-b_lo
        | p_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

RTP Media Congestion Avoidance Techniques
Internet-Draft
Intended status: Experimental
Expires: September 22, 2016

D. Hayes, Ed.
University of Oslo
S. Ferlin
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

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. The signals	3
1.1.1. Packet Loss	3
1.1.2. Packet Delay	3
1.1.3. Path Lag	4
2. Definitions	4
2.1. Parameters and their Effect	7
2.2. Recommended Parameter Values	8
3. Mechanism	8
3.1. SBD feedback requirements	9
3.1.1. Feedback when all the logic is placed at the sender	10
3.1.2. Feedback when the statistics are calculated at the receiver and SBD at the sender	10
3.1.3. Feedback when bottlenecks can be determined at both senders and receivers	11
3.2. Key metrics and their calculation	11
3.2.1. Mean delay	11
3.2.2. Skewness Estimate	11
3.2.3. Variability Estimate	12
3.2.4. Oscillation Estimate	12
3.2.5. Packet loss	13
3.3. Flow Grouping	13
3.3.1. Flow Grouping Algorithm	13
3.3.2. Using the flow group signal	15
3.4. Removing Noise from the Estimates	15
3.4.1. Oscillation noise	15
3.4.2. Clock skew	16
3.5. Reducing lag and Improving Responsiveness	16
3.5.1. Improving the response of the skewness estimate	17
3.5.2. Improving the response of the variability estimate	19
4. Measuring OWD	19
4.1. Time stamp resolution	19
5. Implementation status	20
6. Acknowledgements	20
7. IANA Considerations	20
8. Security Considerations	20

9. Change history	20
10. References	21
10.1. Normative References	21
10.2. Informative References	21
Authors' Addresses	22

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 $M \leq N$.

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 bottleneck determination. It should be large enough to avoid constant switching in the determination, but low enough to ensure that grouping is not attempted when there is no bottleneck and the delay and loss signals cannot be relied upon.
- p_v p_v determines the sensitivity of `freq_est` to noise. Making it smaller will yield higher but noisier values for `freq_est`. Making it too large will render it ineffective for determining groups.
- p_* Flows are separated when the `skew_est|var_est|freq_est` 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 $T=350\text{ms}$, $N=50$, $p_l=0.1$. 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:

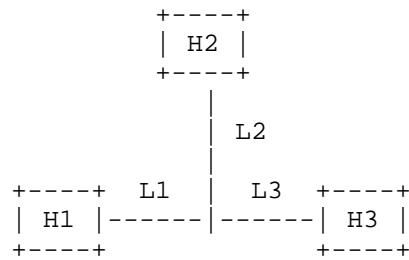
variability (estimate `var_est`, see Section 3.2.3)

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:

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

This 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 $E_T(OWD)$:

$$\text{mean_delay} = E_M(E_T(OWD)) = \text{sum}_M(E_T(OWD)) / M$$

where $M \leq N$. 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`:

$\text{freq_est} = \text{number_of_crossings} / N$

where we define a significant mean crossing as a crossing that extends $p_v * \text{var_est}$ 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 $E_T(\text{OWD})$ a decision is made as to whether this value of $E_T(\text{OWD})$ 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, $\text{number_of_crossings}$, 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 $E_N(E_T(\text{OWD}))$. 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:

$\text{pkt_loss} = \text{sum_NT}(\text{lost packets}) / \text{sum_NT}(\text{total packets})$

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 ($N \cdot T$) 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:

$$\begin{aligned} \text{skew_est} = & ((M-F+1)*\text{sum}(\text{skew_base_T}(1:F)) \\ & + \text{sum}([(M-F):1].*\text{skew_base_T}(F+1:M))) \\ & / ((M-F+1)*\text{sum}(\text{numsampT}(1:F)) \\ & + \text{sum}([(M-F):1].*\text{numsampT}(F+1:M))) \end{aligned}$$

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 initialized to 0

Steps per iteration:

1. old_skewbase = skewbase_hist(M)
2. old_numsampT = numsampT(M)
3. cycle(skewbase_hist)
4. cycle(numsampT)
5. numsampT(1) = num_T(OWD)
6. skewbase_hist(1) = skew_base_T
7. F_skewbase = F_skewbase + skew_base_T - skewbase_hist(F+1)
8. W_D_skewbase = W_D_skewbase + (M-F)*skewbase_hist(F+1) - sum_skewbase
9. W_D_numsamp = W_D_numsamp + (M-F)*numsampT(F+1) - sum_numsamp + F_numsamp
10. F_numsamp = F_numsamp + numsampT(1) - numsampT(F+1)
11. sum_skewbase = sum_skewbase + skewbase_hist(F+1) - old_skewbase
12. sum_numsamp = sum_numsamp + numsampT(1) - old_numsampT
13. skew_est = ((M-F+1)*F_skewbase + W_D_skewbase) / ((M-F+1)*F_numsamp+W_D_numsamp)

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

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

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

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

M. Welzl
K. Hiorth
University of Oslo
March 29, 2018

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 30, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	The Basic Mechanism	3
1.2.	The Signals	3
1.2.1.	Packet Loss	3
1.2.2.	Packet Delay	4
1.2.3.	Path Lag	4
2.	Definitions	4
2.1.	Parameters and Their Effect	6
2.2.	Recommended Parameter Values	7
3.	Mechanism	7
3.1.	SBD Feedback Requirements	8
3.1.1.	Feedback When All the Logic is Placed at the Sender	9
3.1.2.	Feedback When the Statistics are Calculated at the Receiver and SBD Performed at the Sender	9
3.1.3.	Feedback When Bottlenecks can be Determined at Both Senders and Receivers	10
3.2.	Key Metrics and Their Calculation	10
3.2.1.	Mean Delay	10
3.2.2.	Skewness Estimate	11
3.2.3.	Variability Estimate	12
3.2.4.	Oscillation Estimate	12
3.2.5.	Packet Loss	13
3.3.	Flow Grouping	13
3.3.1.	Flow Grouping Algorithm	13
3.3.2.	Using the Flow Group Signal	16
4.	Enhancements to the Basic SBD Algorithm	16
4.1.	Reducing Lag and Improving Responsiveness	16
4.1.1.	Improving the Response of the Skewness Estimate	17
4.1.2.	Improving the Response of the Variability Estimate	19
4.2.	Removing Oscillation Noise	19
5.	Measuring OWD	20
5.1.	Time-stamp Resolution	20
5.2.	Clock Skew	20
6.	Expected Feedback from Experiments	20
7.	Acknowledgments	21
8.	IANA Considerations	21
9.	Security Considerations	21
10.	Change history	21
11.	References	23
11.1.	Normative References	23

11.2. Informative References	23
Authors' Addresses	24

1. Introduction

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

1.1. The Basic Mechanism

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

1.2. The Signals

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

1.2.1. Packet Loss

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

1.2.2. Packet Delay

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

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

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

1.2.3. Path Lag

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

2. Definitions

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

Acronyms used in this document:

OWD -- One Way Delay

MAD -- Mean Absolute Deviation

RTT -- Round Trip Time

SBD -- Shared Bottleneck Detection

Conventions used in this document:

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

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

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

pkt_loss -- a measure of the proportion of packets lost

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

M and F -- number of values related to N

2.1. Parameters and Their Effect

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

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

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

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

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

- c_h** c_h adds hysteresis to the bottleneck determination. It should be large enough to avoid constant switching in the determination, but low enough to ensure that grouping is not attempted when there is no bottleneck and the delay and loss signals cannot be relied upon.
- p_v** p_v determines the sensitivity of freq_est to noise. Making it smaller will yield higher but noisier values for freq_est. Making it too large will render it ineffective for determining groups.
- p_*** Flows are separated when the skew_est|var_est|freq_est|pkt_loss measure is greater than p_s|p_mad|p_f|p_d. Adjusting these is a compromise between false grouping of flows that do not share a bottleneck and false splitting of flows that do. Making them larger can help if the measures are very noisy, but reducing the noise in the statistical measures by adjusting T and N|M may be a better solution.

2.2. Recommended Parameter Values

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

3. Mechanism

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

variability (estimate var_est, see Section 3.2.3)

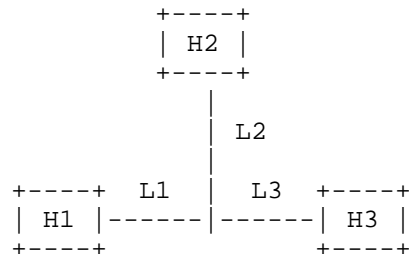
skewness (estimate skew_est, see Section 3.2.2)

oscillation (estimate freq_est, see Section 3.2.4)

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

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

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



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

Figure 1

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

1. Sender-side: consider a situation where host H1 sends media streams to hosts H2 and H3, and L1 is a shared bottleneck. H2 and H3 measure the OWD and packet loss and either send back this raw data, or the calculated summary statistics, periodically to H1 every T. H1, having this knowledge, can determine the shared bottleneck and accordingly control the send rates.
2. Receiver-side: consider that H2 is also sending media to H3, and L3 is a shared bottleneck. If H3 sends summary statistics to H1 and H2, neither H1 nor H2 alone obtain enough knowledge to detect this shared bottleneck; H3 can however determine it by combining the summary statistics related to H1 and H2, respectively.

3.1. SBD Feedback Requirements

There are three possible scenarios each with different feedback requirements:

1. Both summary statistic calculations and SBD are performed at senders only. When sender-based congestion control is implemented, this method is RECOMMENDED.
2. Summary statistics calculated on the receivers and SBD at the senders.

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

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

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

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

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

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

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

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

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

- * The values of T, N, M, and the necessary resolution and precision of the relayed statistics.
- o A response message from the receiver acknowledges this message with a list of key metrics it supports (subset of the senders list) and is able to relay back to the sender.

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

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

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

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

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

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

3.2. Key Metrics and Their Calculation

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

3.2.1. Mean Delay

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

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

$$\text{mean_delay} = E_M(E_T(OWD)) = \text{sum}_M(E_T(OWD)) / M$$

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

3.2.2. Skewness Estimate

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

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

```
if (OWD < mean_delay) skew_base_T++
```

```
if (OWD > mean_delay) skew_base_T--
```

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

$$\text{skew_est} = \text{sum}_{MT}(\text{skew_base_T}) / \text{num}_{MT}(OWD)$$

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:

$$\text{var_base_T} = \text{sum_T}(|\text{OWD} - \text{E_T}(\text{OWD})|)$$

where

$|x|$ is the absolute value of x

$\text{E_T}(\text{OWD})$ is the mean OWD calculated in the previous T

$$\text{var_est} = \text{MAD_MT} = \text{sum_MT}(\text{var_base_T}) / \text{num_MT}(\text{OWD})$$

3.2.4. Oscillation Estimate

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

$$\text{freq_est} = \text{number_of_crossings} / N$$

where we define a significant mean crossing as a crossing that extends $p_v * \text{var_est}$ 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 $\text{E_T}(\text{OWD})$ a decision is made as to whether this value of $\text{E_T}(\text{OWD})$ 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, $\text{number_of_crossings}$, 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 $\text{E_N}(\text{E_T}(\text{OWD}))$. 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:

$$\text{pkt_loss} = \text{sum_NT}(\text{lost packets}) / \text{sum_NT}(\text{total packets})$$

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

3.3. Flow Grouping

3.3.1. Flow Grouping Algorithm

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

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

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

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

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

```
skew_est < c_s
```

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

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

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

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

```
diff(freq_est) < p_f
```

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

```
diff(var_est) < (p_mad * var_est)
```

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

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

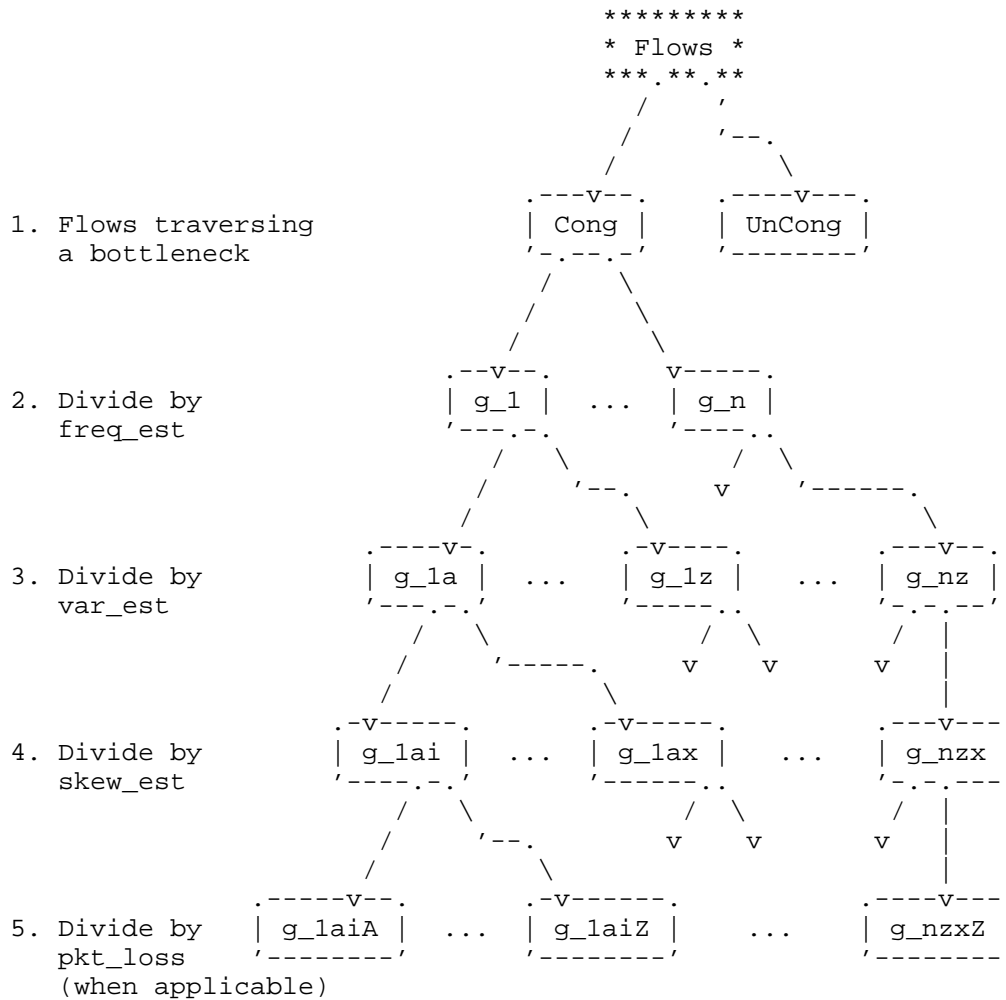
```
diff(skew_est) < p_s
```

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

```
diff(pkt_loss) < (p_d * pkt_loss)
```

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

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



Simple grouping algorithm.

Figure 2

3.3.2. Using the Flow Group Signal

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

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

4. Enhancements to the Basic SBD Algorithm

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

4.1. Reducing Lag and Improving Responsiveness

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

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

1. Reducing N and/or M -- but at the expense of having less accurate metrics, and/or
2. Exploiting the fact that more recent measurements are more valuable than older measurements and weighting them accordingly.

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

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

4.1.1. Improving the Response of the Skewness Estimate

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

$$\begin{aligned} \text{skew_est} = & ((M-F+1)*\text{sum}(\text{skew_base_T}(1:F)) \\ & + \text{sum}([(M-F):1].*\text{skew_base_T}(F+1:M))) \\ & / ((M-F+1)*\text{sum}(\text{numsampT}(1:F)) \\ & + \text{sum}([(M-F):1].*\text{numsampT}(F+1:M))) \end{aligned}$$

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

To calculate this weighted skew_est incrementally:

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

Initialize: sum_skewbase = 0, F_skewbase=0, W_D_skewbase=0
 skewbase_hist = buffer length M initialize to 0
 numsampT = buffer length M initialized to 0

Steps per iteration:

1. old_skewbase = skewbase_hist(M)
2. old_numsampT = numsampT(M)
3. cycle(skewbase_hist)
4. cycle(numsampT)
5. numsampT(1) = num_T(OWD)
6. skewbase_hist(1) = skew_base_T
7. F_skewbase = F_skewbase + skew_base_T - skewbase_hist(F+1)
8. W_D_skewbase = W_D_skewbase + (M-F)*skewbase_hist(F+1) - sum_skewbase
9. W_D_numsamp = W_D_numsamp + (M-F)*numsampT(F+1) - sum_numsamp + F_numsamp
10. F_numsamp = F_numsamp + numsampT(1) - numsampT(F+1)
11. sum_skewbase = sum_skewbase + skewbase_hist(F+1) - old_skewbase
12. sum_numsamp = sum_numsamp + numsampT(1) - old_numsampT
13. skew_est = ((M-F+1)*F_skewbase + W_D_skewbase) / ((M-F+1)*F_numsamp+W_D_numsamp)

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

4.1.2. Improving the Response of the Variability Estimate

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

$$\begin{aligned} \text{var_est} = & ((M-F+1)*\text{sum}(\text{var_base_T}(1:F)) \\ & + \text{sum}([(M-F):1].*\text{var_base_T}(F+1:M))) \\ & / ((M-F+1)*\text{sum}(\text{numsampT}(1:F)) \\ & + \text{sum}([(M-F):1].*\text{numsampT}(F+1:M))) \end{aligned}$$

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

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

4.2. Removing Oscillation Noise

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

To remove this source of noise from `freq_est`:

1. Set the current `var_base_T` = NaN (a value representing an invalid record, i.e. Not a Number) for flows that are deemed to not be transiting a bottleneck by the first `skew_est` based grouping test (see Section 3.3.1).
2. Then `var_est` = `sum_MT(var_base_T != NaN) / num_MT(OWD)`
3. For `freq_est`, only record a significant mean crossing if flow deemed to be transiting a bottleneck.

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

5. Measuring OWD

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

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

5.1. Time-stamp Resolution

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

5.2. Clock Skew

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

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

6. Expected Feedback from Experiments

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

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

7. Acknowledgments

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

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

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

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

10. Change history

XX RFC ED - PLEASE REMOVE THIS SECTION XXX

Changes made to this document:

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

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

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

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

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

clarifications. Figure 2 to supplement grouping algorithm description.

- WG-05->WG-06 : Updates addressing WG reviews
<https://mailarchive.ietf.org/arch/msg/rmcat/-1JdrTMq1Y5T6ZNlOkrQJQ27TzE> and
https://mailarchive.ietf.org/arch/msg/rmcat/eI2Q1f8NL2SxbJgjFLR4_rEmJ_g. This has mainly involved minor clarifications, including the moving of 3.4.1 and 3.5 into the new Section 4, and 3.4.1 into Section 5
- WG-04->WG-05 : Fix ToC formatting. Add section on expected feedback from experiments replacing short section on implementation status. Added comment on ECN as a signal. Clarification of lost packet signaling. Change term "draft" to "document" where appropriate. American spelling. Some tightening of the text.
- WG-03->WG-04 : Add M to terminology table, suggest skew_est based on previous T and no freq_est in clock skew section, feedback requirements as a separate sub section.
- WG-02->WG-03 : Correct misspelled author
- WG-01->WG-02 : Removed ambiguity associated with the term "congestion". Expanded the description of initialization messages. Removed PDV metric. Added description of incremental weighted metric calculations for skew_est. Various clarifications based on implementation work. Fixed typos and tuned parameters.
- WG-00->WG-01 : Moved unbiased skew section to replace skew estimate, more robust variability estimator, the term variance replaced with variability, clock drift term corrected to clock skew, revision to clock skew section with a place holder, description of parameters.
- 02->WG-00 : Fixed missing 0.5 in 3.3.2 and missing brace in 3.3.3
- 01->02 : New section describing improvements to the key metric calculations that help to remove noise, bias, and reduce lag. Some revisions to the

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

00->01 : Revisions to terminology for clarity

11. References

11.1. Normative References

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

11.2. Informative References

- [Hayes-LCN14] Hayes, D., Ferlin, S., and M. Welzl, "Practical Passive Shared Bottleneck Detection using Shape Summary Statistics", Proc. the IEEE Local Computer Networks (LCN) pp150-158, September 2014, <http://heim.ifi.uio.no/davihay/hayes14__pract_passiv_shared_bottl_detec-abstract.html>.
- [I-D.ietf-avtcore-cc-feedback-message] Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", draft-ietf-avtcore-cc-feedback-message-01 (work in progress), March 2018.
- [I-D.ietf-rmcat-coupled-cc] Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-07 (work in progress), September 2017.
- [RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, DOI 10.17487/RFC2680, September 1999, <<https://www.rfc-editor.org/info/rfc2680>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.

- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<https://www.rfc-editor.org/info/rfc4585>>.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <<https://www.rfc-editor.org/info/rfc5124>>.
- [RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <<https://www.rfc-editor.org/info/rfc6817>>.
- [RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <<https://www.rfc-editor.org/info/rfc7679>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [Zhang-Infocom02]
Zhang, L., Liu, Z., and H. Xia, "Clock synchronization algorithms for network measurements", Proc. the IEEE International Conference on Computer Communications (INFOCOM) pp160-169, September 2002, <<http://dx.doi.org/10.1109/INFCOM.2002.1019257>>.

Authors' Addresses

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

Email: davidh@simula.no

Simone Ferlin

Email: simone@ferlin.io

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

Email: michawe@ifi.uio.no

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

Email: kristahi@ifi.uio.no

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 18, 2016

X. Zhu
S. Mena
Cisco Systems
Z. Sarker
Ericsson AB
January 15, 2016

Modeling Video Traffic Sources for RMCAT Evaluations
draft-ietf-rmcat-video-traffic-model-00

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on July 18, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Desired Behavior of A Synthetic Video Traffic Model	3
4. Interactions Between Synthetic Video Traffic Source and Other Components at the Sender	4
5. A Statistical Reference Model	6
5.1. Time-damped response to target rate update	7
5.2. Temporary burst/oscillation during transient	7
5.3. Output rate fluctuation at steady state	8
5.4. Rate range limit imposed by video content	8
6. A Trace-Driven Model	8
6.1. Choosing the video sequence and generating the traces	9
6.2. Using the traces in the syntethic codec	10
6.2.1. Main algorithm	10
6.2.2. Notes to the main algorithm	12
6.3. Varying frame rate and resolution	12
7. Comparing and Combining The Two Models	13
8. Implementation Status	14
9. IANA Considerations	14
10. References	14
10.1. Normative References	14
10.2. Informative References	14
Authors' Addresses	15

1. Introduction

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

On the other hand, evaluation results of a candidate RMCAT algorithm should mostly reflect performance of the congestion control module, and somewhat decouple from 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 RMCAT algorithms using a synthetic video traffic source model that captures key characteristics of the behavior of a live video encoder. To this end, this draft presents two reference models. The first is based on statistical modelling; the second is trace-driven. The draft also discusses the pros and cons of each approach, as well as the possibility to combine both.

2. Terminology

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

3. Desired Behavior of A Synthetic Video Traffic Model

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

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

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

- o To change bitrate. This includes ability to change framerate and/or spatial resolution, or to skip frames when required.
- o To fluctuate around the target bitrate specified by the congestion control module.

- o To delay in convergence to the target bitrate.
- o To generate intra-coded or repair frames on demand.

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

- o 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.
- o Temporal pattern similarity: The individual traffic trace instances generated by the model should mimic the temporal pattern of those from a real video encoder.
- o Statistical resemblance: The synthetic traffic should match the outcome of the real video encoder in terms of statistical characteristics, such as the mean, variance, peak, and autocorrelation coefficients of the bitrate. It is also important that the statistical resemblance should hold across different time scales, ranging from tens of milliseconds to sub-seconds.
- o Wide range of coverage: The model should be easily configurable to cover a wide range of codec behaviors (e.g., with either fast or slow reaction time in live encoder rate control) and video content variations (e.g, ranging from high-motion to low-motion).

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

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

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

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

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

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

- o Target rate $R_v(t)$: requested at time t , typically from the congestion control module. Depending on the congestion control algorithm in use, the update requests can either be periodic (e.g., once per second), or on-demand (e.g., only when a drastic bandwidth change over the network is observed).
- o Target frame rate $FPS(t)$: the instantaneous frame rate measured in frames-per-second at time t . This depends on the native camera capture frame rate as well as the target/preferred frame rate configured by the application or user.
- o Frame resolution $XY(t)$: the 2-dimensional vector indicating the preferred frame resolution in pixels at time t . Several factors govern the resolution requested to the synthetic video encoder over time. Examples of such factors are the capturing resolution of the native camera; or the current target rate $R_v(t)$, since very small resolutions do not make sense with very high bitrates, and vice-versa.
- o 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.
- o On-demand generation of intra (I) frame: the request to encode another I frame to avoid further error propagation at the receiver, if severe packet losses are observed. This request typically comes from the error control module.

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

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

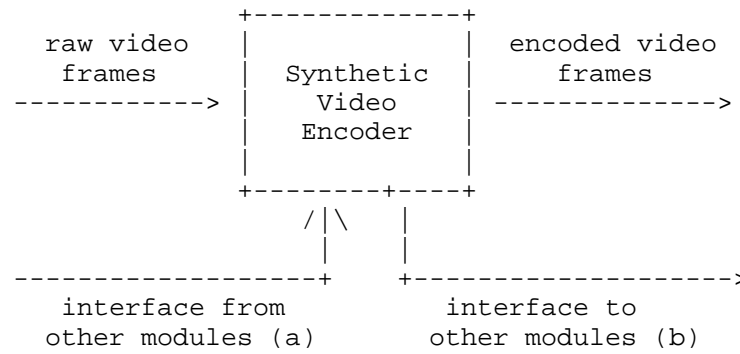


Figure 1: Interaction between synthetic video encoder and other modules at the sender

5. A Statistical Reference Model

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

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

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

5.1. Time-damped response to target rate update

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

5.2. Temporary burst/oscillation during transient

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

This temporary burst is characterized by two parameters:

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

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

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

5.3. Output rate fluctuation at steady state

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

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

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

5.4. Rate range limit imposed by video content

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

6. A Trace-Driven Model

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

The following list summarizes this approach's main steps:

- 1) Choose one or more representative raw video sequences.

- 2) Using an actual live video encoder, encode the sequences at various bitrates. Keep just the sequences of frame sizes for each bitrate.
- 3) Construct a data structure that contains the output of the previous step. The data structure should allow for easy bitrate lookup.
- 4) Upon a target bitrate request $R_v(t)$ from the controller, look up the closest bitrates among those previously stored. Use the frame size sequences stored for those bitrates to approximate the frame sizes to output.
- 5) The output of the synthetic encoder contains "encoded" frames with zeros as contents but with realistic sizes.

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

6.1. Choosing the video sequence and generating the traces

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

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

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

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

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

```

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

```

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

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

6.2. Using the traces in the syntethic codec

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

6.2.1. Main algorithm

We maintain two variables `r_current` and `t_current`:

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

or equal to the new value of $R_v(t)$. For the moment, we assume the value of $R_v(t)$ to be clipped in the range $[R_{\min}, R_{\max}]$.

```

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

```

* t_{current} is an index to the frame size vector stored in $\text{Traces}[r_{\text{current}}]$. It is updated every time a new frame is due. We assume all vectors stored in Traces to have the same size, denoted size_traces . The following equation governs the update of t_{current} :

```

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

```

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

We initialize r_{current} to R_{\min} , and t_{current} to 0.

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

- a) $R_{\min} \leq R_v(t) < R_{\max}$: In this case we use linear interpolation of the frame sizes appearing in $\text{Traces}[r_{\text{current}}]$ and $\text{Traces}[r_{\text{current}} + 1]$. The interpolation is done as follows:

```

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

```

- b) $R_v(t) < R_{\min}$: In this case, we scale the trace sequence with the lowest bitrate, in the following way:

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

c) $R_v(t) \geq R_{\max}$: We also use scaling for this case. We use the trace sequence with the greatest bitrate:

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

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

6.2.2. Notes to the main algorithm

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

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

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

6.3. Varying frame rate and resolution

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

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

improve quality-of-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-based codec to accommodate variable frame rate and/or resolution.

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

7. Comparing and Combining The Two Models

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

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

8. Implementation Status

The statistical model has been implemented as a traffic generator module within the [ns-2] network simulation platform.

More recently, both the statistical and trace-driven models have been implemented as a stand-alone traffic source module. This 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. 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>>.
- [H264] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services", 2003, <<http://www.itu.int/rec/T-REC-H.264-201304-I>>.
- [HEVC] ITU-T Recommendation H.265, "High efficiency video coding", 2015.

10.2. Informative References

- [Hu2010] Hu, H., Ma, Z., and Y. Wang, "Optimization of Spatial, Temporal and Amplitude Resolution for Rate-Constrained Video Coding and Scalable Video Adaptation", in Proc. 19th IEEE International Conference on Image Processing, (ICIP'12), September 2012.
- [Ozer2011] Ozer, J., "Video Compression for Flash, Apple Devices and HTML5", ISBN 13:978-0976259503, 2011.
- [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.

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

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

[Syncodecs]
 Mena, S., D'Aronco, S., and X. Zhu, "Syncodecs: Synthetic
 codecs for evaluation of RMCAT work",
 <<https://github.com/cisco/syncodecs>>.

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: 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.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 23, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Terminology	3
3. Desired Behavior of A Synthetic Video Traffic Model	3
4. Interactions Between Synthetic Video Traffic Source and Other Components at the Sender	5
5. A Statistical Reference Model	6
5.1. Time-damped response to target rate update	7
5.2. Temporary burst and oscillation during the transient period	8
5.3. Output rate fluctuation at steady state	8
5.4. Rate range limit imposed by video content	9
6. A Trace-Driven Model	9
6.1. Choosing the video sequence and generating the traces . .	10
6.2. Using the traces in the synthetic codec	11
6.2.1. Main algorithm	11
6.2.2. Notes to the main algorithm	13
6.3. Varying frame rate and resolution	14
7. Combining The Two Models	14
8. Implementation Status	16
9. IANA Considerations	16
10. Security Considerations	16
11. References	16
11.1. Normative References	16
11.2. Informative References	16
Authors' Addresses	17

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:

- o To change bitrate. This includes the ability to change framerate and/or spatial resolution or to skip frames upon request.
- o To fluctuate around the target bitrate specified by the congestion control module.
- o To show a delay in convergence to the target bitrate.
- o 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.

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

- o 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).
- o 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.
- o 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.

- o 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.
- o 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.

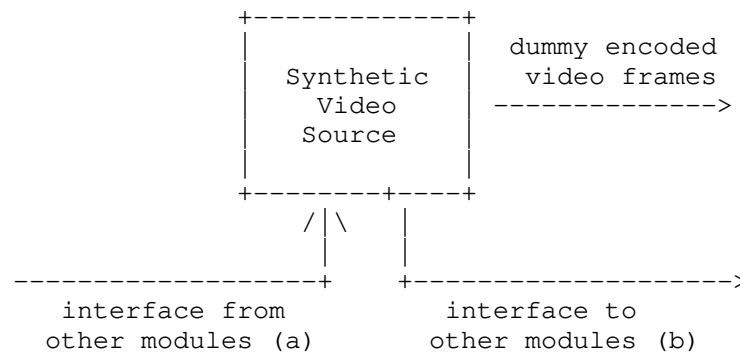


Figure 1: Interaction between synthetic video encoder and other modules at the sender

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 $(t-t_0)/t_0$	0.15
SCALE_B	Scaling parameter of the zero-mean Laplacian distribution describing deviations in normalized frame size $(B-B_0)/B_0$	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:

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

- o Fluctuations in frame interval: the intervals between adjacent frames have been observed to fluctuate around the reference interval of $t_0 = 1/\text{FPS}$. Deviations in normalized frame interval $\text{DELTA}_t = (t-t_0)/t_0$ 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 t_0 .

- o Fluctuations in frame size: the output encoded frame sizes also tend to fluctuate around the reference frame size $B_0 = R_v/8/\text{FPS}$. Likewise, deviations in the normalized frame size $\text{DELTA}_B = (B - B_0)/B_0$ 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 B_0 .

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 an 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} .

- o 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))

```

- o 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 + 1][t_current]
distance_lo = (R_v - r_current) / 1
framesize = size_hi*distance_lo + size_lo*(1-distance_lo)

```

- b) $R_v < R_{f_min}$: the output frame size is calculated via scaling with respect to the lowest bitrate R_{f_min} in the trace file, as follows:

```

w = R_v / R_{f\_min}
framesize = max(fs_min, factor * Traces[R_{f\_min}][t_current])

```

- c) $R_v \geq R_{f_max}$: the output frame size is calculated by scaling with respect to the highest bitrate R_{f_max} in the trace file, as follows:

```

w = R_v / R_{f\_max}
framesize = min(fs_max, w * Traces[R_{f\_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:

- o 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.
- o 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.

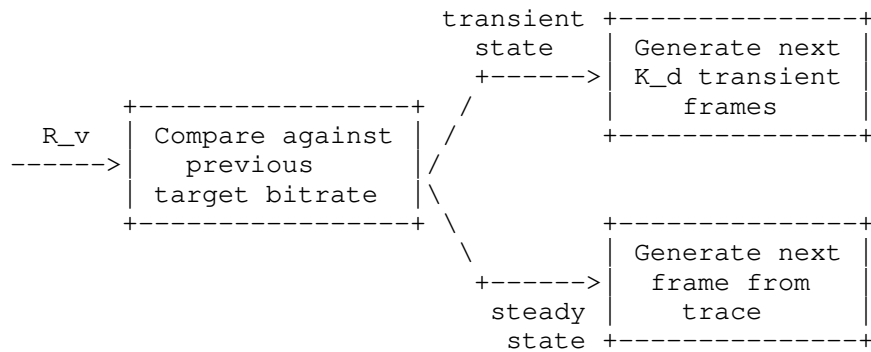


Figure 3: A hybrid video traffic model

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 τ_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: May 8, 2016

Z. Sarker
I. Johansson
Ericsson AB
X. Zhu
J. Fu
W. Tan
M. Ramalho
Cisco Systems
November 5, 2015

Evaluation Test Cases for Interactive Real-Time Media over Wireless
Networks
draft-ietf-rmcat-wireless-tests-01

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 May 8, 2016.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminologies	3
3. Cellular Network Specific Test Cases	3
3.1. Varying Network Load	6
3.1.1. Network Connection	6
3.1.2. Simulation Setup	7
3.2. Bad Radio Coverage	8
3.2.1. Network connection	9
3.2.2. Simulation Setup	9
3.3. Desired Evaluation Metrics for cellular test cases	10
4. Wi-Fi Networks Specific Test Cases	10
4.1. Bottleneck in Wired Network	12
4.1.1. Network topology	12
4.1.2. Test setup	13
4.1.3. Typical test scenarios	14
4.1.4. Expected behavior	14
4.2. Bottleneck in Wi-Fi Network	14
4.2.1. Network topology	15
4.2.2. Test setup	15
4.2.3. Typical test scenarios	16
4.2.4. Expected behavior	17
4.3. Potential Potential Test Cases	17
4.3.1. EDCA/WMM usage	17
4.3.2. Legacy 802.11b Effects	17
5. Conclusion	18
6. Acknowledgements	18
7. IANA Considerations	18
8. Security Considerations	18
9. References	18
9.1. Normative References	18
9.2. Informative References	19

Authors' Addresses	20
--------------------	----

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 -

- o 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.
- o Queues are always per radio bearer hence each user can have many of such queues.
- o Users can experience both Inter and Intra Radio Access Technology (RAT) handovers ("handover" definition in [HO-def-3GPP]).
- o 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.
- o The network part decides how much the user can transmit.
- o 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.
- o 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

- o Shared and varying link capacity
- o Mobility
- o 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.

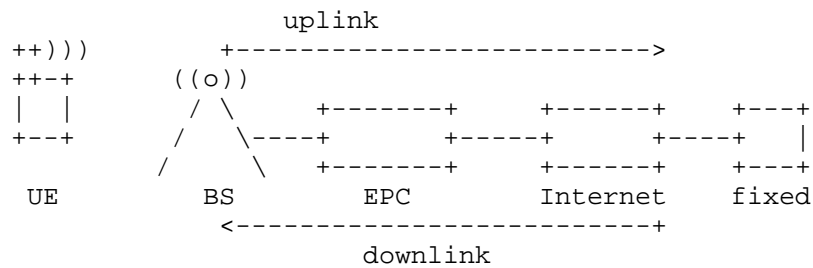


Figure 1: Simulation Topology

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-

- o Average cell throughput (for all cells), shows cell utilizations.
- o Application sending and receiving bitrate, goodput.
- o Packet Loss Rate (PLR).
- o End to end Media frame delay. For video, this means the delay from capture to display.
- o Transport delay.
- o 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:

- o The wireless radio channel is subject to interference from nearby transmitters, multi-path fading, and shadowing, causing fluctuations in link throughput and sometimes an error-prone communication environment
- o Available network bandwidth is not only shared over the air between concurrent users, but also between uplink and downlink traffic due to the half duplex nature of wireless transmission medium.
- o Packet transmissions 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.

- o 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 signal strength. A different choice of Physical-layer rate will lead to different application-layer throughput.
- o Presence of legacy 802.11b networks can significantly slow down 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.]
- o Handover from one Wi-Fi Access Point (AP) to another may cause packet delay and loss.
- o 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 802.11g due to its wide availability in network simulation platform. In practice, however, statistics collected from enterprise 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. Whenever possible, it is recommended to extend some of the experiments to 802.11n and 802.11ac, so as to reflect a more modern Wi-Fi network setting.

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

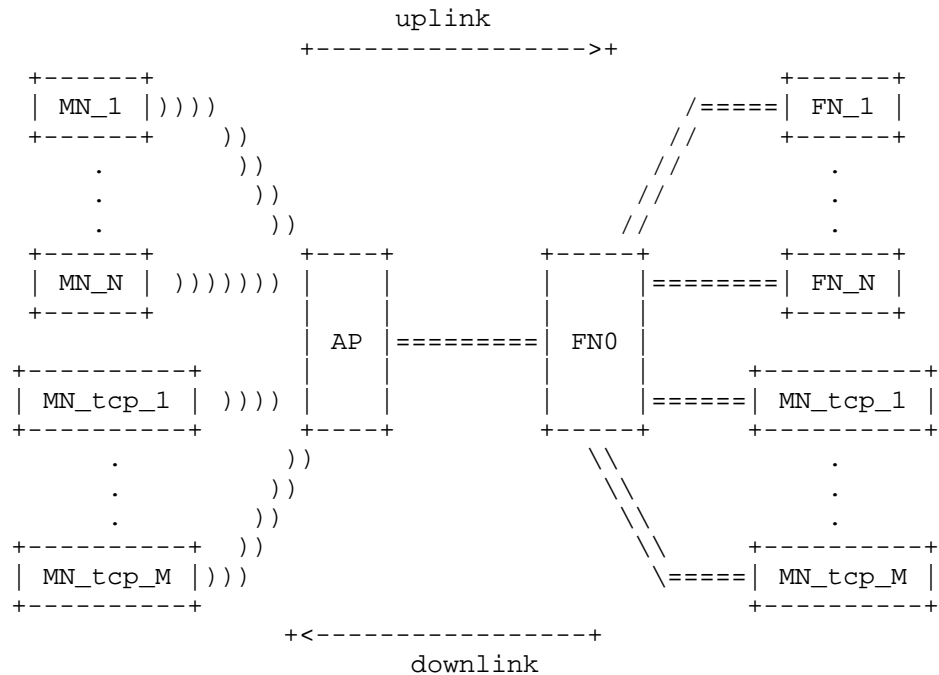


Figure 2: Network topology for Wi-Fi test cases

4.1.2. Test setup

- o Test duration: 120s
- o Wi-Fi network characteristics:
 - * Radio propagation model: Log-distance path loss propagation model [NS3WiFi]
 - * PHY- and MAC-layer configuration: IEEE 802.11g
 - * PHY-layer link rate: 54 Mbps
- o 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%.
- o 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
 - + Traffic direction: See Section 4.1.3

- + Number of sources (M): See Section 4.1.3
- + Congestion control: Default TCP congestion control [TBD]
- + Traffic timeline:
 - Start time: 0s
 - End time: 119s

4.1.3. Typical test scenarios

- o Single uplink RMCAT flow: N=1 with uplink direction and M=0.
- o One pair of bi-directional RMCAT flows: N=2 (with one uplink flow and one downlink flow); M=0.
- o One RMCAT flow competing against one long-live TCP flow over uplink: N=1 (uplink) and M = 1(uplink).

4.1.4. Expected behavior

- o 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 excessive rate oscillations.
- o 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 of the presence of measurement noise over the Wi-Fi connection.
- o One RMCAT flow competing with long-live TCP flow over uplink: the candidate algorithm should be able to avoid congestion collapse, and stabilize 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

- o Test duration: 120s
- o Wi-Fi network characteristics:
 - * Radio propagation model: Log-distance path loss propagation model [NS3WiFi]
 - * PHY- and MAC-layer configuration: IEEE 802.11g
 - * PHY-layer link rate: 54 Mbps
- o 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%.
- o 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
- + Number of sources (M): See Section 4.2.3
- + Traffic direction: See Section 4.2.3
- + Congestion control: Default TCP congestion control [TBD]
- + Traffic timeline:
 - Start time: 0s
 - End time: 119s

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.

- o 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 802.11g with a physical-layer transmission rate of 54 Mbps is chosen. Note that retransmission and MAC-layer headers and control packets may be sent at a lower link speed. The total application-layer throughput (reasonable distance, low interference and small number of contention stations) for 802.11g is around 20 Mbps. Consequently, a total of 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 stabilize at a fair share of bandwidth.
- o 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 stabilize at a fair share of bandwidth.
- o 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.

- o 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. [Editor's Note: more detailed description will be added in the next version in terms of directoin/number of RMCAT and TCP flows.]
- o 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

- o 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.
- o 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).
- o Multiple bi-directional 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). 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 legacy 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]
Singh, V. and J. Ott, "Evaluating Congestion Control for Interactive Real-time Media", draft-ietf-rmcat-eval-criteria-04 (work in progress), October 2015.
- [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., Singh, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-ietf-rmcat-eval-test-02 (work in progress), September 2015.
- [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
Laboratoriegärend 11
Luleå 97753
Sweden

Phone: +46 107173743
Email: zaheduzzaman.sarker@ericsson.com

Ingemar Johansson
Ericsson AB
Laboratoriegärend 11
Luleå 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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 14, 2020

Z. Sarker
Ericsson AB
X. Zhu
J. Fu
Cisco Systems
March 13, 2020

Evaluation Test Cases for Interactive Real-Time Media over Wireless
Networks
draft-ietf-rmcat-wireless-tests-11

Abstract

The Real-time Transport Protocol (RTP) is a common transport choice for interactive multimedia communication applications. The performance of these applications typically depends on a well-functioning congestion control algorithm. To ensure a seamless and robust user experience, a well-designed RTP-based congestion control algorithm should work well across all access network types. This document describes test cases for evaluating performances of candidate congestion control algorithms over cellular and Wi-Fi networks.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 14, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	3
2. Cellular Network Specific Test Cases	3
2.1. Varying Network Load	6
2.1.1. Network Connection	6
2.1.2. Simulation Setup	7
2.1.3. Expected behavior	9
2.2. Bad Radio Coverage	9
2.2.1. Network connection	9
2.2.2. Simulation Setup	9
2.2.3. Expected behavior	10
2.3. Desired Evaluation Metrics for cellular test cases	10
3. Wi-Fi Networks Specific Test Cases	10
3.1. Bottleneck in Wired Network	12
3.1.1. Network topology	12
3.1.2. Test/simulation setup	13
3.1.3. Typical test scenarios	14
3.1.4. Expected behavior	15
3.2. Bottleneck in Wi-Fi Network	15
3.2.1. Network topology	15
3.2.2. Test/simulation setup	16
3.2.3. Typical test scenarios	17
3.2.4. Expected behavior	18
3.3. Other Potential Test Cases	19
3.3.1. EDCA/WMM usage	19
3.3.2. Effect of heterogeneous link rates	19
4. IANA Considerations	20
5. Security Considerations	20
6. Contributors	20
7. Acknowledgments	21
8. References	21
8.1. Normative References	21
8.2. Informative References	22
Authors' Addresses	22

1. Introduction

Wireless networks (both cellular and Wi-Fi [IEEE802.11]) are an integral and increasingly more significant part of the Internet. Typical application scenarios for interactive multimedia communication over wireless include from video conferencing calls in a bus or train as well as live media streaming at home. It is well known that the characteristics and technical challenges for supporting multimedia services over wireless are very different from those of providing the same service over a wired network. Although the basic test cases as defined in [I-D.ietf-rmcat-eval-test] have covered many common effects of network impairments for evaluating RTP-based congestion control schemes, they remain to be tested over characteristics and dynamics unique to a given wireless environment. For example, in cellular networks, the base station maintains individual queues per radio bearer per user hence it leads to a different nature of interactions between traffic flows of different users. This contrasts with a typical wired network setting where traffic flows from all users share the same queue at the bottleneck. Furthermore, user mobility patterns in a cellular network differ from those in a Wi-Fi network. Therefore, it is important to evaluate the performance of proposed candidate RTP-based congestion control solutions over cellular mobile networks and over Wi-Fi networks respectively.

The draft [I-D.ietf-rmcat-eval-criteria] provides the guideline for evaluating candidate algorithms and recognizes the importance of testing over wireless access networks. However, it does not describe any specific test cases for performance evaluation of candidate algorithms. This document describes test cases specifically targeting cellular and Wi-Fi networks.

2. Cellular Network Specific Test Cases

A cellular environment is more complicated than its wireline counterpart 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 radio links will often act as a bottleneck for the rest of the network and 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 remain some packet losses and delay variations. Moreover, with increased cell load or handover to a congested cell, congestion in the transport network will become even worse. Besides, there exist certain characteristics that distinguish the cellular network

from other wireless access networks such as Wi-Fi. In a cellular network --

- o 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.
 - * Leftover/unused resources can be consumed by other greedy users.
- o Queues are always per radio bearer hence each user can have many such queues.
- o Users can experience both Inter and Intra Radio Access Technology (RAT) handovers (see [HO-def-3GPP] for the definition of "handover").
- o Handover between cells or change of serving cells (as described 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.
- o The network part decides how much the user can transmit.
- o The cellular network has variable link capacity per user.
 - * It can vary as fast as a period of milliseconds.
 - * It depends on many factors (such as distance, speed, interference, different flows).
 - * It uses complex and smart link adaptation which makes the link behavior ever more dynamic.
 - * The scheduling priority depends on the estimated throughput.
- o Both Quality of Service (QoS) and non-QoS radio bearers can be used.

Hence, a real-time communication application operating over a cellular network needs to cope with a shared bottleneck link and variable link capacity, events like handover, non-congestion related loss, abrupt changes in bandwidth (both short term and long term) due to handover, network load and bad radio coverage. Even though 3GPP

has defined QoS bearers [QoS-3GPP] to ensure high-quality user experience, it is still preferable for real-time applications to behave in an adaptive manner.

Different mobile operators deploy their own cellular networks with their own set of network functionalities and policies. Usually, a mobile operator network includes a range of radio access technologies such as 3G and 4G/LTE. Looking at the specifications of such radio technologies it is evident that only the more recent 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 for such genre of application.

The key factors in defining test cases for cellular networks are:

- o Shared and varying link capacity
- o Mobility
- o Handover

However, these factors are typically highly correlated in a cellular network. Therefore, instead of devising separate test cases for individual important events, we have divided the test case into two categories. It should be noted that the goal of the following test cases is to evaluate the performance of candidate algorithms over the 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 introduce any extra congestion along the path. Consequently, this draft has kept as out of scope the combination of multiple access technologies involving both cellular and Wi-Fi users. In this latter case the shared bottleneck is likely at the wired backhaul link. These test cases further assume a typical real-time telephony scenario where one real-time session consists of one voice stream and one video stream.

Even though it is possible to carry out tests over operational cellular networks (e.g., LTE/5G), and actually such tests are already available today, these tests cannot in general be carried out in a deterministic fashion to ensure repeatability. The main reason is that these networks are controlled by cellular operators and there exist various amounts of competing traffic in the same cell(s). In practice, it is only in underground mines that one can carry out near deterministic testing. Even there, it is not guaranteed either as workers in the mines may carry with them their personal mobile phones. Furthermore, the underground mining setting may not reflect

typical usage patterns in an urban setting. We, therefore, recommend that a cellular network simulator is used for the test cases defined in this document, for example -- the LTE simulator in [NS-3].

2.1. 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 user/UE in the media session is an endpoint following RTP-based congestion control. User arrivals follow a Poisson distribution proportional to the length of the call, to keep the number of users per cell fairly constant during the evaluation period. At the beginning of the simulation, there should be enough 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 should be excluded from the evaluation period. Typically, the evaluation period starts 30 seconds after test initialization.

This test case also includes user mobility and some competing traffic. The latter includes both the same types of flows (with same adaptation algorithms) and different types of flows (with different services and congestion control schemes).

2.1.1. Network Connection

Each mobile user is connected to a fixed user. The connection between the mobile user and fixed user consists of a cellular radio access, an Evolved Packet Core (EPC) and an Internet connection. The mobile user is connected to the EPC using cellular radio access technology which is further connected to the Internet. At the other end, the fixed user is connected to the Internet via wired connection with sufficiently high bandwidth, for instance, 10 Gbps, so that the system bottleneck is on the cellular radio access interface. The wired connection to in this setup does not introduce any network impairments to the test; it only adds 10 ms of one-way propagation delay.

The path from the fixed user to the mobile users is defined as "Downlink" and the path from the mobile users to the fixed user is defined as "Uplink". We assume that only uplink or downlink is congested for mobile users. Hence, we recommend that the uplink and downlink simulations are run separately.

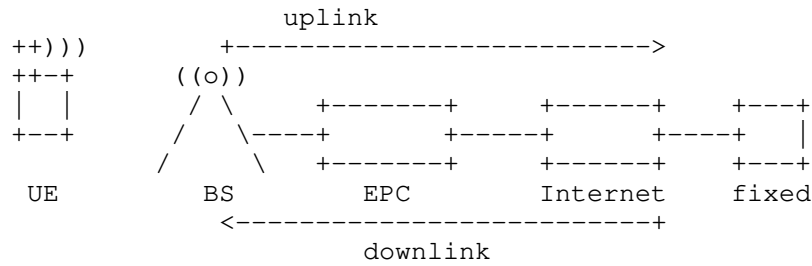


Figure 1: Simulation Topology

2.1.2. Simulation Setup

The values enclosed within "[]" for the following simulation attributes follow the same notion as in [I-D.ietf-rmcat-eval-test]. The desired simulation setup is as follows --

1. Radio environment:
 - A. Deployment and propagation model: 3GPP case 1 (see [HO-deploy-3GPP])
 - B. Antenna: Multiple-Input and Multiple-Output (MIMO), 2D or 3D antenna pattern.
 - 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): [40ms, 150ms]
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 intensity : {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 defined 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 Bit Rate (CBR)
 - e. Media bitrate: 20 Kbps
 - f. Adaptation: off
 8. Other traffic models:
 - * Downlink simulation: Maximum of 4Mbps/cell (web browsing or FTP traffic following default TCP congestion control [RFC5681])
 - * Uplink simulation: Maximum of 2Mbps/cell (web browsing or FTP traffic following default TCP congestion control [RFC5681])

2.1.3. Expected behavior

The investigated congestion control algorithms should result in 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 levels.

2.2. 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 a larger cell radius than that in the previous test case. In this test case, each user/UE in the media session is an endpoint following RTP-based congestion control. User arrivals follow a Poisson distribution proportional to the length of the call, to keep the number of users per cell 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 should be excluded from the evaluation period. Typically, the evaluation period starts 30 seconds after test initialization.

This test case also includes user mobility and some competing traffic. The latter includes the same kind of flows (with same adaptation algorithms).

2.2.1. Network connection

Same as defined in Section 2.1.1

2.2.2. Simulation Setup

The desired simulation setup is the same as the Varying Network Load test case defined in Section 2.1 except the following changes:

1. Radio environment: Same as defined in Section 2.1.2 except the following:
 - A. Deployment and propagation model: 3GPP case 3 (see [HO-deploy-3GPP])
 - 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 2.1.2
4. Other traffic models:
 - * Downlink simulation: Maximum of 2Mbps/cell (web browsing or FTP traffic following default TCP congestion control [RFC5681])
 - * Unlink simulation: Maximum of 1Mbps/cell (web browsing or FTP traffic following default TCP congestion control [RFC5681])

2.2.3. Expected behavior

The investigated congestion control algorithms should result in 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 levels.

2.3. Desired Evaluation Metrics for cellular test cases

The evaluation criteria document [I-D.ietf-rmcat-eval-criteria] defines the metrics to be used to evaluate candidate algorithms. Considering the nature and distinction of cellular networks we recommend that at least the following metrics be used to evaluate the performance of the candidate algorithms:

- o Average cell throughput (for all cells), shows cell utilizations.
- o Application sending and receiving bitrate, goodput.
- o Packet Loss Rate (PLR).
- o End-to-end Media frame delay. For video, this means the delay from capture to display.
- o Transport delay.
- o Algorithm stability in terms of rate variation.

3. Wi-Fi Networks Specific Test Cases

Given the prevalence of Internet access links over Wi-Fi, it is important to evaluate candidate RTP-based congestion control solutions over test cases that include Wi-Fi access links. Such

evaluations should highlight the inherently different characteristics of Wi-Fi networks in contrast to their wired counterparts:

- o The wireless radio channel is subject to interference from nearby transmitters, multipath fading, and shadowing. These effects lead to fluctuations in the link throughput and sometimes an error-prone communication environment.
- o Available network bandwidth is not only shared over the air between concurrent users but also between uplink and downlink traffic due to the half-duplex nature of the wireless transmission medium.
- o Packet transmissions 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 over the air and significant network overhead, as well as packet drops due to buffer overflow at the transmitters. This, in turn, leads to excessive delay, retransmissions, packet losses and lower effective bandwidth for applications. Note further that the collision-induced delay and loss patterns are qualitatively different from those caused by congestion over a wired connection.
- o The IEEE 802.11 standard (i.e., Wi-Fi) supports multi-rate transmission capabilities by dynamically choosing the most appropriate modulation and coding scheme (MCS) for the given received signal strength. A different choice in the MCS Index leads to different physical-layer (PHY-layer) link rates and consequently different application-layer throughput.
- o The presence of legacy devices (e.g., ones operating only in IEEE 802.11b) at a much lower PHY-layer link rate can significantly slow down the rest of a modern Wi-Fi network. As discussed in [Heusse2003], the main reason for such anomaly is that it takes much longer to transmit the same packet over a slower link than over a faster link, thereby consuming a substantial portion of air time.
- o Handover from one Wi-Fi Access Point (AP) to another may lead to excessive packet delays and losses during the process.
- o IEEE 802.11e has introduced the Enhanced Distributed Channel Access (EDCA) mechanism to allow different traffic categories to contend for channel access using different random back-off parameters. This mechanism is a mandatory requirement for the Wi-Fi Multimedia (WMM) certification in Wi-Fi Alliance. It allows for prioritization of real-time application traffic such as voice

and video over non-urgent data transmissions (e.g., file transfer).

In summary, the presence of Wi-Fi access links in different network topologies can exert different impact on the network performance in terms of application-layer effective throughput, packet loss rate, and packet delivery delay. These, in turn, will influence the behavior of end-to-end real-time multimedia congestion control.

Unless otherwise mentioned, the test cases in this section choose the PHY- and MAC-layer parameters based on the IEEE 802.11n Standard. Statistics collected from enterprise Wi-Fi networks show that the two dominant physical modes are 802.11n and 802.11ac, accounting for 41% and 58% of connected devices. As Wi-Fi standards evolve over time -- for instance, with the introduction of the emerging Wi-Fi 6 (based on IEEE 802.11ax) products -- the PHY- and MAC-layer test case specifications need to be updated accordingly to reflect such changes.

Typically, a Wi-Fi access network connects to a wired infrastructure. Either the wired or the Wi-Fi segment of the network can be the bottleneck. The following sections 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.

We recommend to carry out the test cases as defined in this document using a simulator, such as [NS-2] or [NS-3]. When feasible, it is encouraged to perform testbed-based evaluations using Wi-Fi access points and endpoints running up-to-date IEEE 802.11 protocols, such as 802.11ac and the emerging Wi-Fi 6, so as to verify the viability of the candidate schemes.

3.1. Bottleneck in Wired Network

The test scenarios below are intended to mimic the setup 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 as in [I-D.ietf-rmcat-eval-test], it is still worthwhile to run through these tests as sanity checks.

3.1.1. Network topology

Figure 2 shows the network topology of 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 a RTP-based media flow

or a TCP traffic flow. Directions of the flows can be uplink (i.e., from mobile nodes to fixed nodes), downlink (i.e., from fixed nodes to mobile nodes), or bi-directional. The total number of uplink/downlink/bi-directional flows for RTP-based media traffic and TCP traffic are denoted as N and M, respectively.

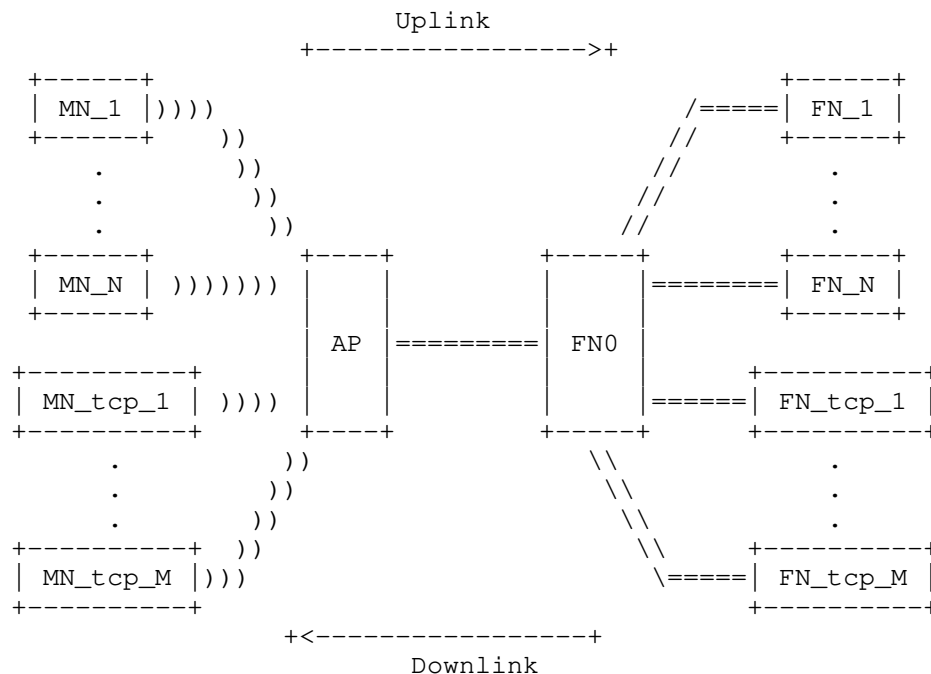


Figure 2: Network topology for Wi-Fi test cases

3.1.2. Test/simulation setup

- o Test duration: 120s
- o Wi-Fi network characteristics:
 - * Radio propagation model: Log-distance path loss propagation model (see [NS3WiFi])
 - * PHY- and MAC-layer configuration: IEEE 802.11n
 - * MCS Index at 11: 16-QAM 1/2, Raw Data Rate at 52Mbps
- o 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%.
- o Application characteristics:
 - * Media Traffic:
 - + Media type: Video
 - + Media direction: See Section 3.1.3
 - + Number of media sources (N): See Section 3.1.3
 - + Media timeline:
 - Start time: 0s.
 - End time: 119s.
 - * Competing traffic:
 - + Type of sources: long-lived TCP or CBR over UDP
 - + Traffic direction: See Section 3.1.3
 - + Number of sources (M): See Section 3.1.3
 - + Congestion control: Default TCP congestion control [RFC5681] or constant-bit-rate (CBR) traffic over UDP.
 - + Traffic timeline: See Section 3.1.3

3.1.3. Typical test scenarios

- o Single uplink RTP-based media flow: N=1 with uplink direction and M=0.
- o One pair of bi-directional RTP-based media flows: N=2 (i.e., one uplink flow and one downlink flow); M=0.

- o One pair of bi-directional RTP-based media flows: N=2; one uplink on-off CBR flow over UDP: M=1 (uplink). The CBR flow has ON time at t=0s-60s and OFF time at t=60s-119s.
- o One pair of bi-directional RTP-based media flows: N=2; one uplink off-on CBR flow over UDP: M=1 (uplink). The CBR flow has OFF time at t=0s-60s and ON time at t=60s-119s.
- o One RTP-based media flow competing against one long-live TCP flow in the uplink direction: N=1 (uplink) and M = 1(uplink). The TCP flow has start time at t=0s and end time at t=119s.

3.1.4. Expected behavior

- o Single uplink RTP-based media flow: the candidate algorithm is expected to detect the path capacity constraint, to converge to the bottleneck link capacity, and to adapt the flow to avoid unwanted oscillations when the sending bit rate is approaching the bottleneck link capacity. No excessive oscillations in the media rate should be present.
- o Bi-directional RTP-based media flows: the candidate algorithm is expected to converge to the bottleneck capacity of the wired path in both directions despite the presence of measurement noise over the Wi-Fi connection. In the presence of background TCP or CBR over UDP traffic, the rate of RTP-based media flows should adapt promptly to the arrival and departure of background traffic flows.
- o One RTP-based media flow competing with long-live TCP flow in the uplink direction: the candidate algorithm is expected to avoid congestion collapse and to stabilize at a fair share of the bottleneck link capacity.

3.2. Bottleneck in Wi-Fi Network

The test cases in this section assume that the wired segment along the media path is well-provisioned whereas the bottleneck exists over the Wi-Fi access network. This is to mimic the application scenarios typically encountered by users in an enterprise environment or at a coffee house.

3.2.1. Network topology

Same as defined in Section 3.1.1

3.2.2. Test/simulation setup

- o Test duration: 120s
- o Wi-Fi network characteristics:
 - * Radio propagation model: Log-distance path loss propagation model (see [NS3WiFi])
 - * PHY- and MAC-layer configuration: IEEE 802.11n
 - * MCS Index at 11: 16-QAM 1/2, Raw Data Rate at 52Mbps
- o 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%.
- o Application characteristics:
 - * Media Traffic:
 - + Media type: Video
 - + Media direction: See Section 3.2.3.
 - + Number of media sources (N): See Section 3.2.3.
 - + 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 3.2.3.

- + Traffic direction: See Section 3.2.3.
- + Congestion control: Default TCP congestion control [RFC5681] or constant-bit-rate (CBR) traffic over UDP.
- + Traffic timeline: See Section 3.2.3.

3.2.3. Typical test scenarios

This section describes a few test scenarios that are deemed as important for understanding the behavior of a candidate RTP-based congestion control scheme over a Wi-Fi network.

- a. Multiple RTP-based media flows sharing the wireless downlink: $N=16$ (all downlink); $M = 0$. This test case is for studying the impact of contention on the multiple concurrent media flows. For an 802.11n network, given the MCS Index of 11 and the corresponding link rate of 52Mbps, the total application-layer throughput (assuming reasonable distance, low interference and infrequent contentions caused by competing streams) is around 20Mbps. A total of $N=16$ RTP-based media flows (with a maximum rate of 1.5Mbps each) are expected to saturate the wireless interface in this experiment. Evaluation of a given candidate scheme should focus on whether the downlink media flows can stabilize at a fair share of the total application-layer throughput.
- b. Multiple RTP-based media flows sharing the wireless uplink: $N = 16$ (all uplink); $M = 0$. When multiple clients attempt to transmit media packets uplink over the Wi-Fi network, they introduce more frequent contentions and potential collisions. Per-flow throughput is expected to be lower than that in the previous downlink-only scenario. Evaluation of a given candidate scheme should focus on whether the uplink flows can stabilize at a fair share of the total application-layer throughput.
- c. Multiple bi-directional RTP-based media flows: $N = 16$ (8 uplink and 8 downlink); $M = 0$. The goal of this test is to evaluate the performance of the candidate scheme in terms of bandwidth fairness between uplink and downlink flows.
- d. Multiple bi-directional RTP-based media flows with on-off CBR traffic over UDP: $N = 16$ (8 uplink and 8 downlink); $M = 5$ (uplink). The goal of this test is to evaluate the adaptation behavior of the candidate scheme when its available bandwidth changes due to the departure of background traffic. The background traffic consists of several (e.g., $M=5$) CBR flows

transported over UDP. These background flows are ON at time $t=0-60s$ and OFF at time $t=61-120s$.

- e. Multiple bi-directional RTP-based media flows with off-on CBR traffic over UDP: $N = 16$ (8 uplink and 8 downlink); $M = 5$ (uplink). The goal of this test is to evaluate the adaptation behavior of the candidate scheme when its available bandwidth changes due to the arrival of background traffic. The background traffic consists of several (e.g., $M=5$) parallel CBR flows transported over UDP. These background flows are OFF at time $t=0-60s$ and ON at times $t=61-120s$.
- f. Multiple bi-directional RTP-based media flows in the presence of background TCP traffic: $N=16$ (8 uplink and 8 downlink); $M = 5$ (uplink). The goal of this test is to evaluate how RTP-based media flows compete against TCP over a congested Wi-Fi network for a given candidate scheme. TCP flows have start time at $t=40s$ and end time at $t=80s$.
- g. Varying number of RTP-based media flows: A series of tests can be carried out for the above test cases with different values of N , e.g., $N = [4, 8, 12, 16, 20]$. The goal of this test is to evaluate how a candidate scheme responds to varying traffic load/demand over a congested Wi-Fi network. The start times of the media flows are randomly distributed within a window of $t=0-10s$; their end times are randomly distributed within a window of $t=110-120s$.

3.2.4. Expected behavior

- o Multiple downlink RTP-based media flows: each media flow is expected to get its fair share of the total bottleneck link bandwidth. Overall bandwidth usage should not be significantly lower than that experienced by the same number of concurrent downlink TCP flows. In other words, the behavior of multiple concurrent TCP flows will be used as a performance benchmark for this test scenario. The end-to-end delay and packet loss ratio experienced by each flow should be within an acceptable range for real-time multimedia applications.
- o Multiple uplink RTP-based media flows: overall bandwidth usage by all media flows should not be significantly lower than that experienced by the same number of concurrent uplink TCP flows. In other words, the behavior of multiple concurrent TCP flows will be used as a performance benchmark for this test scenario.
- o Multiple bi-directional RTP-based media flows with dynamic background traffic carrying CBR flows over UDP: the media flows

are expected to adapt in a timely fashion to the changes in available bandwidth introduced by the arrival/departure of background traffic.

- o Multiple bi-directional RTP-based media flows with dynamic background traffic over TCP: during the presence of TCP background flows, the overall bandwidth usage by all media flows should not be significantly lower than those achieved by the same number of bi-directional TCP flows. In other words, the behavior of multiple concurrent TCP flows will be used as a performance benchmark for this test scenario. All downlink media flows are expected to obtain similar bandwidth as each other. The throughput of each media flow is expected to decrease upon the arrival of TCP background traffic and, conversely, increase upon their departure. Both reactions should occur in a timely fashion, for example, within 10s of seconds.
- o Varying number of bi-directional RTP-based media flows: the test results for varying values of N -- while keeping all other parameters constant -- is expected to show steady and stable per-flow throughput for each value of N. The average throughput of all media flows is expected to stay constant around the maximum rate when N is small, then gradually decrease with increasing value of N till it reaches the minimum allowed rate, beyond which the offered load to the Wi-Fi network exceeds its capacity (i.e., with a very large value of N).

3.3. Other Potential Test Cases

3.3.1. EDCA/WMM usage

The EDCA/WMM mechanism defines prioritized QoS for four traffic classes (or Access Categories). RTP-based real-time media flows should achieve better performance in terms of lower delay and fewer packet losses with EDCA/WMM enabled when competing against non-interactive background traffic such as file transfers. When most of the traffic over Wi-Fi is dominated by media, however, turning on WMM may degrade performance since all media flows now attempt to access the wireless transmission medium more aggressively, thereby causing more frequent collisions and collision-induced losses. This is a topic worthy of further investigation.

3.3.2. Effect of heterogeneous link rates

As discussed in [Heusse2003], the presence of clients operating over slow PHY-layer link rates (e.g., a legacy 802.11b device) connected to a modern network may adversely impact the overall performance of the network. Additional test cases can be devised to evaluate the

effect of clients with heterogeneous link rates on the performance of the candidate congestion control algorithm. Such test cases, for instance, can specify that the PHY-layer link rates for all clients span over a wide range (e.g., 2Mbps to 54Mbps) for investigating its effect on the congestion control behavior of the real-time interactive applications.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

The security considerations in [I-D.ietf-rmcat-eval-criteria] and the relevant congestion control algorithms apply. The principles for congestion control are described in [RFC2914], and in particular, any new method must implement safeguards to avoid congestion collapse of the Internet.

Given the difficulty of deterministic wireless testing, it is recommended and expected that the tests described in this document would be done via simulations. However, in the case where these test cases are carried out in a testbed setting, the evaluation should take place in a controlled lab environment. In the testbed, the applications, simulators and network nodes ought to be well-behaved and should not impact the desired results. It is important to take appropriate caution to avoid leaking non-responsive traffic with unproven congestion avoidance behavior onto the open Internet.

6. Contributors

The following individuals contributed to the design, implementation, and verification of the proposed test cases during earlier stages of this work. They have helped to validate and substantially improve this specification.

Ingemar Johansson, <ingemar.s.johansson@ericsson.com> of Ericsson AB contributing to the description and validation of cellular test cases during the earlier stage of this draft.

Wei-Tian Tan, <dtan2@cisco.com>, of Cisco Systems designed and set up a Wi-Fi testbed for evaluating parallel video conferencing streams, based upon which proposed Wi-Fi test cases are described. He also recommended additional test cases to consider, such as the impact of EDCA/WMM usage.

Michael A. Ramalho, <mar42@cornell.edu> of AcousticComms Consulting (previously at Cisco Systems) applied learnings from Cisco's internal

experimentation to the early versions of the draft. He also worked on validating the proposed test cases in a VM-based lab setting.

7. Acknowledgments

The authors would like to thank Tomas Frankkila, Magnus Westerlund, Kristofer Sandlund, Sergio Mena de la Cruz, and Mirja Kuehlewind for their valuable inputs and review comments regarding this draft.

8. References

8.1. Normative References

- [HO-deploy-3GPP]
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>.
- [I-D.ietf-rmcat-eval-criteria]
Singh, V., Ott, J., and S. Holmer, "Evaluating Congestion Control for Interactive Real-time Media", draft-ietf-rmcat-eval-criteria-13 (work in progress), March 2020.
- [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.
- [IEEE802.11]
IEEE, "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.
- [NS3WiFi] "Wi-Fi Channel Model in ns-3 Simulator", <https://www.nsnam.org/doxygen/classns3_1_1_yans_wifi_channel.html>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<https://www.rfc-editor.org/info/rfc5681>>.
- [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>>.

8.2. Informative References

[Heusse2003]

Heusse, M., Rousseau, F., Berger-Sabbatel, G., and A. Duda, "Performance anomaly of 802.11b", in Proc. 23th Annual Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM'03), March 2003.

[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>.

[NS-2]

"ns-2", December 2014, <http://nsnam.sourceforge.net/wiki/index.php/Main_Page>.

[NS-3]

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

[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>.

[RFC2914]

Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Laboratoriegården 11
Luleå 97753
Sweden

Phone: +46 107173743
Email: zaheduzzaman.sarker@ericsson.com

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

Email: xiaoqzhu@cisco.com

Jiantao Fu
Cisco Systems
771 Alder Drive
Milpitas, CA 95035
USA

Email: jianfu@cisco.com