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

Table of Contents

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.

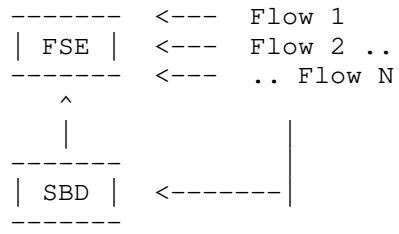

Figure 1: Coupled congestion control architecture

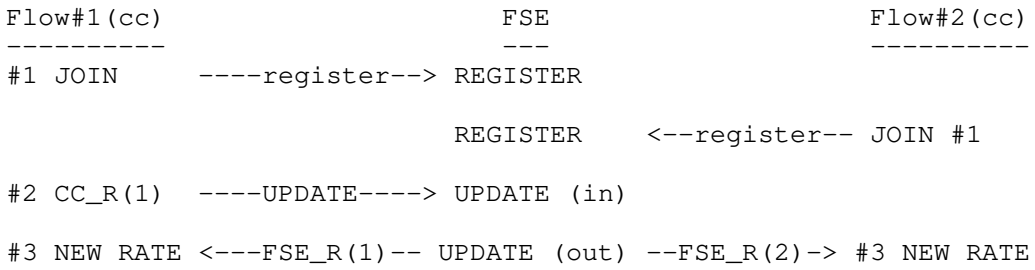

Figure 2: Flow-FSE interaction

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:

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

- 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.

- CC_R(f) - The rate received from the congestion controller of flow f when it calls UPDATE.

- FSE_R(f) - The rate calculated by the FSE for flow f.

- DR(f) - The desired rate of flow f.

- S_CR - The sum of the calculated rates of all flows in the same FG; this value is used to calculate the sending rate.
- FG - A group of flows having the same FGI, and hence sharing the same bottleneck.
- 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).
- S_P - The sum of all the priorities.
- TLO - The total leftover rate: the sum of rates that could not be assigned to flows that were limited by their desired rate.
- 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 DELTA, 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 Adapation (NADA) [I-D.ietf-rmcat-nada] is a congestion control scheme for rtcweb. It calculates a reference rate r_ref upon receiving an acknowledgment, and then, based on the reference rate, it calculates a video target rate r_vin and a sending rate for the flows, r_send.

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

6.2. 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 Stiemerling, 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://safiquli.at.ifi.uio.no/paper/fse-tech-report.pdf , 2014.

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

[I-D.ietf-rmcat-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) = 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(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) = infinity.
3 a) new_S_CR = 5.33; DELTA = 1.
3 b) FSE_R(2) = 4.33. DELTA is positive, hence S_CR = 12;
     DR(2) = 4.33.
3 c) S_P = 1.5.
3 d) Rate(2) = min(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) = infinity.
3 a) new_S_CR = 11.33; DELTA = -2.
3 b) FSE_R(2) = 7.33. DELTA 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(infinity, 0.5/0.5*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

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

D.1.2. Changes from -01 to -02

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

D.1.3. Changes from -02 to -03

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

D.1.4. Changes from -03 to -04

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

D.1.5. Changes from -04 to -05

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

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

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

- Moved scheduling section to the appendix.

D.2.2. Changes from -00 to -01

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

D.2.3. Changes from -01 to -02

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

D.2.4. Changes from -02 to -03

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

D.2.5. Changes from -03 to -04

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

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

D.2.6. Changes from -04 to -05

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

D.2.7. Changes from -05 to -06

- Incorporated comments by Colin Perkins.

D.2.8. Changes from -06 to -07

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

D.2.9. Changes from -07 to -08

- 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

- 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

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

Table of Contents

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.

- Define the test case:
  - General description: describes the motivation and the goals of the test case.
  - Expected behavior: describes the desired rate adaptation behavior.
  - Define a list of metrics to evaluate the desired behavior: this indicates the minimum set of metrics (e.g., link utilization, media sending rate) that a proposed algorithm needs to measure to validate the expected rate adaptation behavior. It should also indicate the time granularity (e.g., averaged over 10ms, 100ms, or 1s) for measuring certain metrics. Typical measurement interval is 200ms.
- Define testbed topology: every test case needs to define an evaluation testbed topology. Figure 1 shows such an evaluation topology. In this evaluation topology, S1..Sn are traffic sources. These sources generate media traffic and use 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.

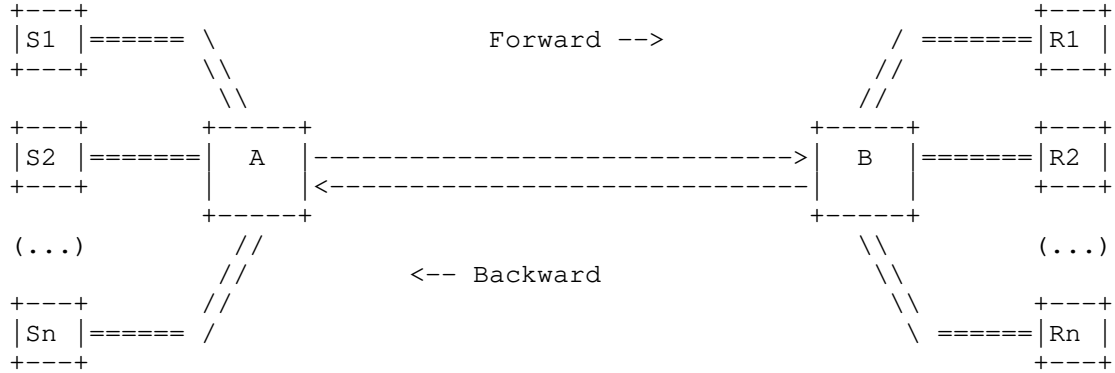

Figure 1: Example of A Testbed Topology

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.

- 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 souce [I-D.ietf-rmcat-video-traffic-model] is used, then the synthetic video traffic source needs to configure 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.

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

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

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

- Bottleneck queue size: 300ms.

- Path loss ratio: 0%.

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

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

4.3. Media source

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

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

range typical of consumer-grade video conferencing applications in designing the test case. If a different bit rate range is used in the test cases, then the end-to-end path capacity values will also need to be scaled accordingly.

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

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

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

+ Responsiveness to new bit rate request: 100ms

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

- an intermediate bottleneck
- change in available capacity (e.g., due to interface change, routing change, abrupt arrival/departure of background non-adaptive traffic).
- maximum media bit rate is greater than link capacity. In this case, 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.

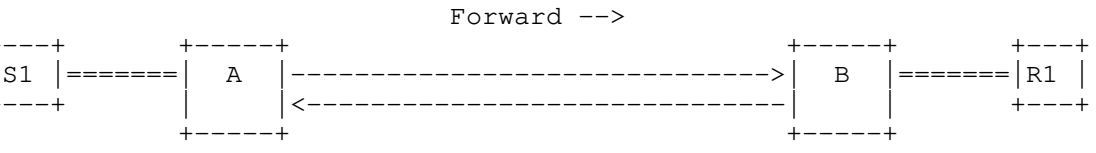

Figure 2: Testbed Topology for Limited Link Capacity

Testbed attributes:

- Test duration: 100s
- Path characteristics: as described in Section 4.2
- Application-related:
  - Media Traffic:
    - Media type: Video
      - Media direction: forward.
      - Number of media sources: one (1)
      - Media timeline:
        - Start time: 0s.
        - End time: 99s.
    - Media type: Audio
      - Media direction: forward.
      - Number of media sources: one (1)
      - Media timeline:
        - Start time: 0s.
        - End time: 99s.
  - Competing traffic:
    - Number of sources : zero (0)
- Test Specific Information:
  - One-way propagation delay: [50 ms, 100 ms]. on the forward path direction
  - This test uses bottleneck path capacity variation as listed in Table 1
  - When using background non-adaptive UDP traffic to induce time-varying bottleneck , the physical path capacity remains at 4Mbps and the UDP traffic source rate changes over time as (4 -

(Y x 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.

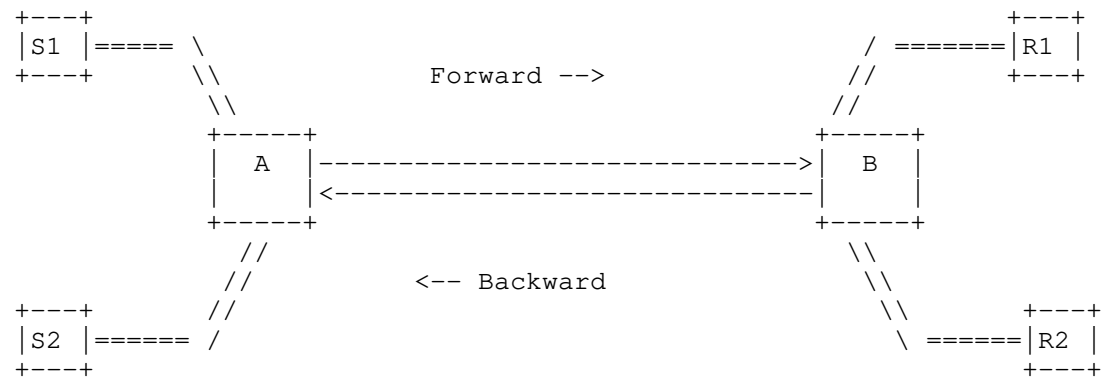

Figure 3: Testbed Topology for Variable Available Capacity

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 x 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.

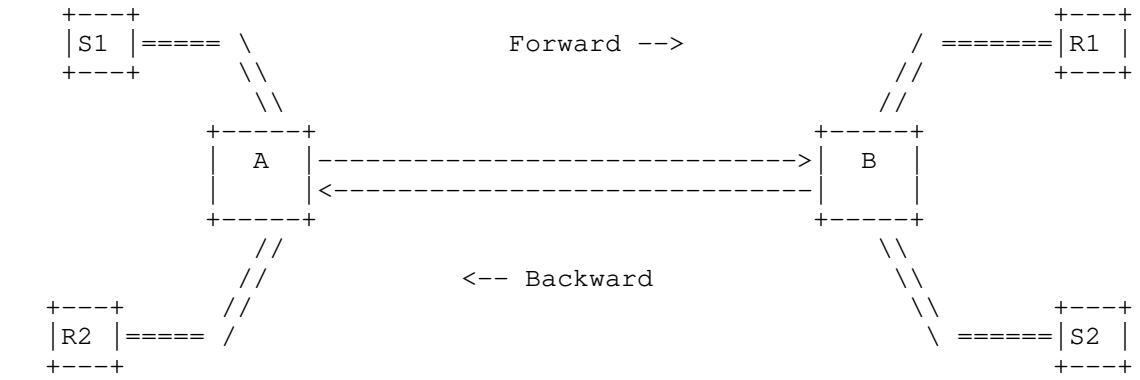

Figure 4: Testbed Topology for Congested Feedback Link

Testbed attributes:

- Test duration: 100s
- Path characteristics:

* Reference bottleneck capacity: 1Mbps.

- Application-related:

  * Media Source:

    + Media type: Video

      - Media direction: forward and backward

      - Number of media sources: two (2)

      - Media timeline:

        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)

- 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.

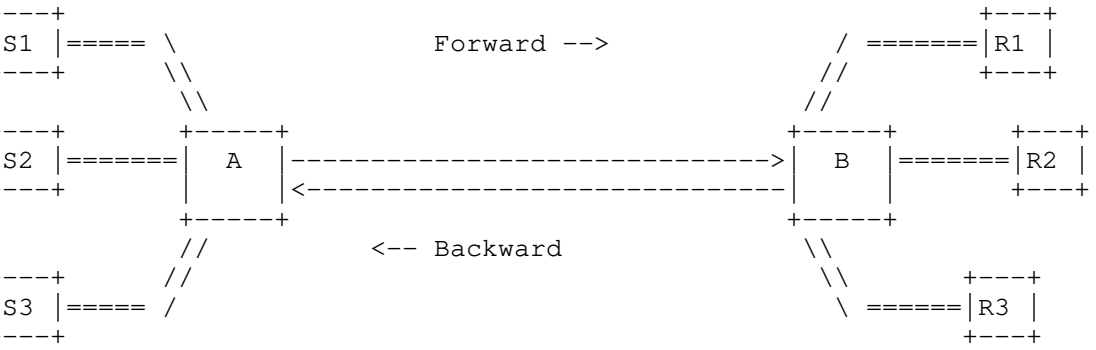

Figure 5: Testbed Topology for Multiple congestion controlled media Flows

Testbed attributes:

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

- Media direction: forward.

- Number of media sources: three (3)

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

* Competing traffic:

+ Number of sources : zero (0)

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:

- Test duration: 300s
- 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.
- 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)
- Test Specific Information: Table 6 defines the media timeline for both media type.

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

Table 6: Media Timeline for Video and Audio media sources

5.6. Media Flow Competing with a Long TCP Flow

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

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

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

1. Flow level:

    A. TCP throughput.

    B. Loss for the TCP flow

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

over the forward path from, S_tcp with acknowledgment packets go over the backward path from, R_tcp.

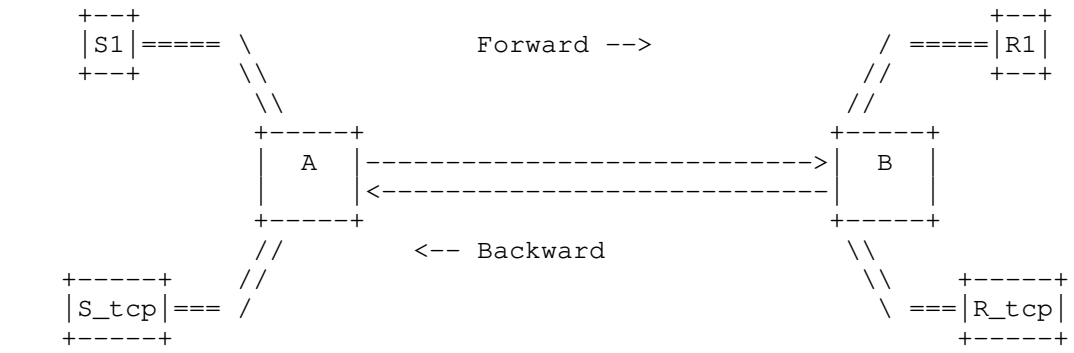

Figure 6: Testbed Topology for TCP vs congestion controlled media Flows

Testbed attributes:

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

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

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

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

- Number of media sources: two (2)
- Media timeline:
  - Start time: 5s.
  - End time: 299s.

* Competing traffic:
  + Number and Types of sources : ten (10), short-lived TCP flows.
  + Traffic direction : forward
  + Congestion algorithm: default TCP Congestion control [RFC5681]. 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.

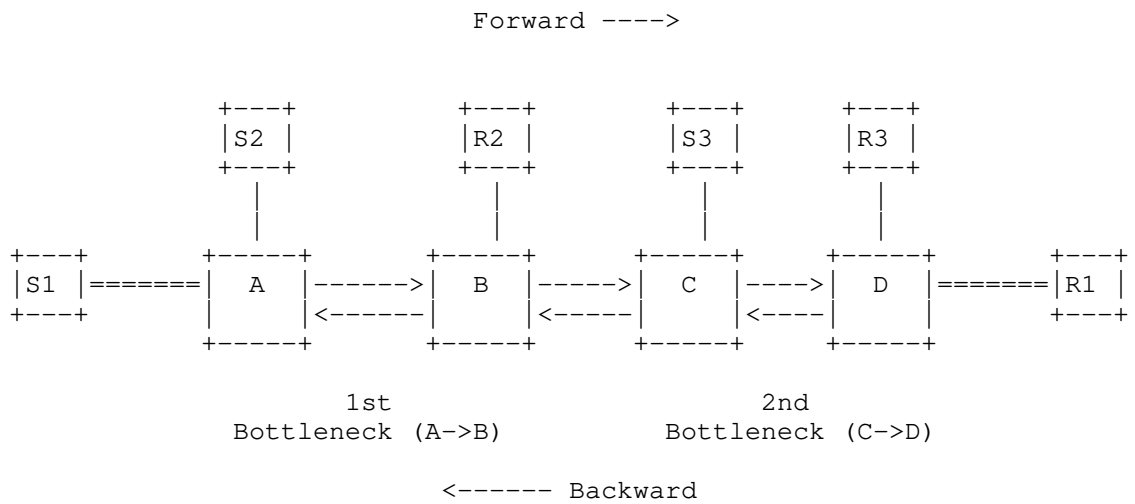

Figure 7: Testbed Topology for Multiple Bottlenecks

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:

- Test duration: 300s
- Path characteristics:
  - Reference bottleneck capacity: 2Mbps.
  - Path capacity ratio between A and B: 1.0
  - Path capacity ratio between B and C: 4.0.
  - Path capacity ratio between C and D: 0.75.
  - One-Way propagation delay:
    1. Between S1 and R1: 100ms
    2. Between S2 and R2: 40ms
    3. Between S3 and R3: 40ms
- Application-related:
  - Media Source:
    - Media type: Video
      - Media direction: Forward
      - Number of media sources: Three (3)
      - Media timeline:
        - Start time: 0s.
        - End time: 299s.
    - Media type: Audio
      - Media direction: Forward
      - Number of media sources: Three (3)
      - Media timeline:
        - Start time: 0s.
        - 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: Informational
Expires: August 23, 2019

X. Zhu
S. Mena
Cisco Systems
Z. Sarker
Ericsson AB

February 19, 2019

Video Traffic Models for RTP Congestion Control Evaluations
draft-ietf-rmcat-video-traffic-model-07

Abstract

This document describes two reference video traffic models for evaluating RTP congestion control algorithms. The first model statistically characterizes the behavior of a live video encoder in response to changing requests on the target video rate. The second model is trace-driven and emulates the output of actual encoded video frame sizes from a high-resolution test sequence. Both models are designed to strike a balance between simplicity, repeatability, and authenticity in modeling the interactions between a live video traffic source and the congestion control module. Finally, the document describes how both approaches can be combined into a hybrid model.

Table of Contents

1. Introduction

When evaluating candidate congestion control algorithms designed for real-time interactive media, it is important to account for the characteristics of traffic patterns generated from a live video encoder. Unlike synthetic traffic sources that can conform perfectly to the rate changing requests from the congestion control module, a live video encoder can be sluggish in reacting to such changes. The output rate of a live video encoder also typically deviates from the target rate due to uncertainties in the encoder rate control process.

Consequently, end-to-end delay and loss performance of a real-time media flow can be further impacted by rate variations introduced by the live encoder.

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

One way to strike a balance between the above considerations is to evaluate congestion control algorithms using a synthetic video traffic source model that captures key characteristics of the behavior of a live video encoder. The synthetic traffic model should also contain tunable parameters so that it can be flexibly adjusted to reflect the wide variations in real-world live video encoder behaviors. To this end, this draft presents two reference models. The first is based on statistical modeling. The second is driven by frame size and interval traces recorded from a real-world encoder. The draft also discusses the pros and cons of each approach, as well as how both approaches can be combined into a hybrid model.

2. Terminology

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

3. Desired Behavior of A Synthetic Video Traffic Model

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

Due to uncertainties in the captured video scene, the output rate typically deviates from the specified target. In the presence of a significant change in target rate, the encoder’s output frame sizes sometimes fluctuate for a short, transient period of time before the output rate converges to the new target. Finally, while most of the frames in a live session are encoded in predictive mode (i.e., P-frames in [H264]), the encoder can occasionally generate a large intra-coded frame (i.e., I-frame as defined in [H264]) or a frame

partially containing intra-coded blocks in an attempt to recover from losses, to re-sync with the receiver, or during the transient period of responding to target rate or spatial resolution changes.

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

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

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

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

These distinct behavior features can be characterized via simple statistical modeling or a trace-driven approach. Section 5 and Section 6 provide an example of each approach, respectively. Section 7 discusses how both models can be combined together.

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

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

The synthetic video source dynamically generates a sequence of dummy video frames with varying size and interval. These dummy frames are processed by other modules in order to transmit the video stream over the network. During the lifetime of a video transmission session, the synthetic video source will typically be required to adapt its encoding bitrate, and sometimes the spatial resolution and frame rate.

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

- Target bitrate R_v: target bitrate request measured in bits per second (bps). Typically, the congestion control module calculates the target bitrate and updates it dynamically over time. Depending on the congestion control algorithm in use, the update requests can either be periodic (e.g., once per second), or on-demand (e.g., only when a drastic bandwidth change over the network is observed).

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

- Target frame resolution XY: the 2-dimensional vector indicating the preferred frame resolution in pixels. Several factors govern the resolution requested to the synthetic video source over time. Examples of such factors include the capturing resolution of the native camera and the display size of the destination screen. The target frame resolution also depends on the current target bitrate R_v, since it does not make sense to pair very low spatial resolutions with very high bitrates, and vice-versa.

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

- On-demand generation of intra (I) frame: the request to encode another I frame to avoid further error propagation at the receiver when severe packet losses are observed. This request typically comes from the error control module. It can be initiated either by the sender or by the receiver via Full Intra Request (FIR) messages as defined in [RFC5104].

An example of outgoing interface call --- marked as (b) in Figure 1 --- is the rate range [R_min, R_max]. Here, R_min and R_max are meant to capture the dynamic rate range and actual live video encoder is capable of generating given the input video content. This typically depends on the video content complexity and/or display type (e.g., higher R_max for video contents with higher motion complexity, or for displays of higher resolution). Therefore, these values will not change with R_v but may change over time if the content is changing.

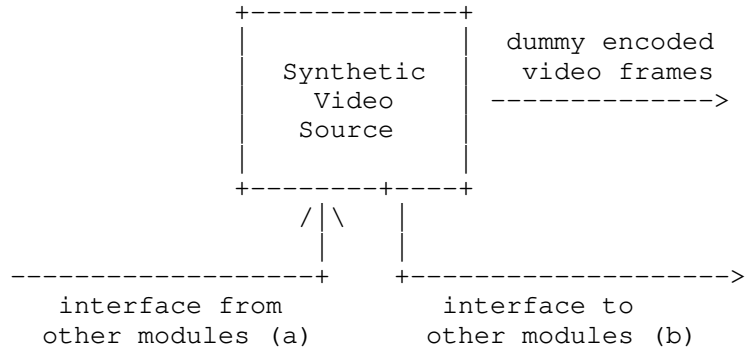

Figure 1: Interaction between synthetic video encoder and other modules at the sender

5. A Statistical Reference Model

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

Notation	Parameter Name	Example Value
R_v	Target bitrate request	1 Mbps
FPS	Target frame rate	30 Hz
tau_v	Encoder reaction latency	0.2 s
K_d	Burst duration of the transient period	8 frames
K_B	Burst frame size during the transient period	13.5 KBytes*
t0	Reference frame interval 1/FPS	33 ms
B0	Reference frame size R_v/8/FPS	4.17 KBytes
SCALE_t	Scaling parameter of the zero-mean Laplacian distribution describing deviations in normalized frame interval (t-t0)/t0	0.15
SCALE_B	Scaling parameter of the zero-mean Laplacian distribution describing deviations in normalized frame size (B-B0)/B0	0.15
R_min	minimum rate supported by video encoder type or content activity	150 Kbps
R_max	maximum rate supported by video encoder type or content activity	1.5 Mbps

* Example value of K_B for a video stream encoded at 720p and 30 frames per second, using H.264/AVC encoder.

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

5.1. Time-damped response to target rate update

While the congestion control module can update its target bitrate request R_v at any time, the statistical model dictates that the encoder will only react to such changes tau_v seconds after a

previous rate transition. In other words, when the encoder has reacted to a rate change request at time t, it will simply ignore all subsequent rate change requests until time t+tau_v.

5.2. Temporary burst and oscillation during the transient period

The output bitrate R_o during the period [t, t+tau_v] is considered to be in a transient state when reacting to abrupt changes in target rate. Based on observations from video encoder output data, the encoder reaction to a new target bitrate request can be characterized by high variations in output frame sizes. It is assumed in the model that the overall average output bitrate R_o during this transient period matches the target bitrate R_v. Consequently, the occasional burst of large frames is followed by smaller-than-average encoded frames.

This temporary burst is characterized by two parameters:

- burst duration K_d: number of frames in the burst event; and
- burst frame size K_B: size of the initial burst frame which is typically significantly larger than average frame size at steady state.

It can be noted that these burst parameters can also be used to mimic the insertion of a large on-demand I frame in the presence of severe packet losses. The values of K_d and K_B typically depend on the type of video codec, spatial and temporal resolution of the encoded stream, as well as the video content activity level.

5.3. Output rate fluctuation at steady state

The output bitrate R_o during steady state is modeled as randomly fluctuating around the target bitrate R_v. The output traffic can be characterized as the combination of two random processes denoting the frame interval t and output frame size B over time, as the two major sources of variations in the encoder output. For simplicity, the deviations of t and B from their respective reference levels are modeled as independent and identically distributed (i.i.d) random variables following the Laplacian distribution [Papoulis]. More specifically:

- Fluctuations in frame interval: the intervals between adjacent frames have been observed to fluctuate around the reference interval of t0 = 1/FPS. Deviations in normalized frame interval DELTA_t = (t-t0)/t0 can be modeled by a zero-mean Laplacian distribution with scaling parameter SCALE_t. The value of SCALE_t dictates the "width" of the Laplacian distribution and therefore

the amount of fluctuation in actual frame intervals (t) with respect to the reference frame interval t0.

- Fluctuations in frame size: the output encoded frame sizes also tend to fluctuate around the reference frame size B0=R_v/8/FPS. Likewise, deviations in the normalized frame size DELTA_B = (B-B0)/B0 can be modeled by a zero-mean Laplacian distribution with scaling parameter SCALE_B. The value of SCALE_B dictates the "width" of this second Laplacian distribution and correspondingly the amount of fluctuations in output frame sizes (B) with respect to the reference target B0.

Both values of SCALE_t and SCALE_B can be obtained via parameter fitting from empirical data captured for a given video encoder. Example values are listed in Figure 2 based on empirical data presented in [IETF-Interim].

5.4. Rate range limit imposed by video content

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

6. A Trace-Driven Model

The second approach for modeling a video traffic source is trace-driven. This can be achieved by running an actual live video encoder on a set of chosen raw video sequences and using the encoder's output traces for constructing a synthetic video source. With this approach, the recorded video traces naturally exhibit temporal fluctuations around a given target bitrate request R_v from the congestion control module.

The following list summarizes the main steps of this approach:

1. Choose one or more representative raw video sequences.

2. Encode the sequence(s) using an actual live video encoder. Repeat the process for a number of bitrates. Keep only the sequence of frame sizes for each bitrate.

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

4. Upon a target bitrate request R_v from the controller, look up the closest bitrates among those previously stored. Use the

frame size sequences stored for those bitrates to approximate the frame sizes to output.

5. The output of the synthetic video traffic source contains "encoded" frames with dummy contents but with realistic sizes.

In the following, Section 6.1 explains the first three steps (1-3), Section 6.2 elaborates on the remaining two steps (4-5). Finally, Section 6.3 briefly discusses the possibility to extend the trace-driven model for supporting time-varying frame rate and/or time-varying frame resolution.

6.1. Choosing the video sequence and generating the traces

The first step is a careful choice of a set of video sequences that are representative of the target use cases for the video traffic model. For the example use case of interactive video conferencing, it is recommended to choose a sequence with content that resembles a "talking head", e.g. from a news broadcast or recording of an actual video conferencing call.

The length of the chosen video sequence is a tradeoff. If it is too long, it will be difficult to manage the data structures containing the traces. If it is too short, there will be an obvious periodic pattern in the output frame sizes, leading to biased results when evaluating congestion control performance. It has been empirically determined that a sequence 2 to 4 minutes in length sufficiently avoids the periodic pattern.

Given the chosen raw video sequence, denoted S, one can use a live encoder, e.g. some implementation of [H264] or [HEVC], to produce a set of encoded sequences. As discussed in Section 3, the output bitrate of the live encoder can be achieved by tuning three input parameters: quantization step size, frame rate, and picture resolution. In order to simplify the choice of these parameters for a given target rate, one can typically assume a fixed frame rate (e.g. 30 fps) and a fixed resolution (e.g., 720p) when configuring the live encoder. See Section 6.3 for a discussion on how to relax these assumptions.

Following these simplifications, the chosen encoder can be configured to start at a constant target bitrate, then vary the quantization step size (internally via the video encoder rate controller) to meet various externally specified target rates. It can be further assumed the first frame is encoded as an I-frame and the rest are P-frames (see, e.g., [H264] for definitions of I- and P-frames). For live encoding, the encoder rate control algorithm typically does not use knowledge of frames in the future when encoding a given frame.

Given the minimum and maximum bitrates at which the synthetic codec is to operate (denoted as R_min and R_max, see Section 4), the entire range of target bitrates can be divided into n_s steps. This leads to a encoding bitrate ladder of (n_s + 1) choices equally spaced apart by the step length l = (R_max - R_min)/n_s. The following simple algorithm is used to encode the raw video sequence.

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

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

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

Finally, note that, as mentioned in previous sections, R_min and R_max may be modified after the initial sequences are encoded. Henceforth, for notational clarity, we refer to the bitrate range of the trace file as [Rf_min, Rf_max]. The algorithm described in the next section also covers the cases when the current target bitrate is less than Rf_min, or greater than Rf_max.

6.2. Using the traces in the synthetic codec

The main idea behind the trace-driven synthetic codec is that it mimics the rate adaptation behavior of a real live codec upon dynamic updates of the target bitrate request R_v by the congestion control module. It does so by switching to a different frame size vector stored in the map Traces when needed.

6.2.1. Main algorithm

The main algorithm for rate adaptation in the synthetic codec maintains two variables: r_current and t_current.

- The variable r_current points to one of the keys of map Traces. Upon a change in the value of R_v, typically because the congestion controller detects that the network conditions have changed, r_current is updated based on R_v as follows:

```
R_ref = min (Rf_max, max(Rf_min, R_v))

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

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

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

where operator % denotes modulo, and SkipFrames is a predefined constant that denotes the number of frames to be skipped at the beginning of frame size vectors after t_current has wrapped around. The point of constant SkipFrames is avoiding the effect of periodically sending a large I-frame followed by several smaller-than-average P-frames. A typical value of SkipFrames is 20, although it could be set to 0 if one is interested in studying the effect of sending I-frames periodically.

The initial value of r_current is set to R_min, and the initial value of t_current is set to 0.

When a new frame is due, its size can be calculated following one of the three cases below:

a) Rf_min <= R_v < Rf_max: the output frame size is calculated via linear interpolation of the frame sizes appearing in Traces[r_current] and Traces[r_current + 1]. The interpolation is done as follows:

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

b) R_v < Rf_min: the output frame size is calculated via scaling with respect to the lowest bitrate Rf_min in the trace file, as follows:

```
w = R_v / Rf_min
framesize = max(fs_min, factor * Traces[Rf_min][t_current])
```

c) R_v >= Rf_max: the output frame size is calculated by scaling with respect to the highest bitrate Rf_max in the trace file, as follows:

```
w = R_v / Rf_max
framesize = min(fs_max, w * Traces[Rf_max][t_current])
```

In cases b) and c), floating-point arithmetic is used for computing the scaling factor w. The resulting value of the instantaneous frame size (framesize) is further clipped within a reasonable range between fs_min (e.g., 10 bytes) and fs_max (e.g., 1MB).

6.2.2. Notes to the main algorithm

Note that the main algorithm as described above can be further extended to mimic some additional typical behaviors of a live video encoder. Two examples are given below:

- I-frames on demand: The synthetic codec can be extended to simulate the sending of I-frames on demand, e.g., as a reaction to losses. To implement this extension, the codec's incoming interface (see (a) in Figure 1) is augmented with a new function to request a new I-frame. Upon calling such function, t_current is reset to 0.

- Variable step length l between R_min and R_max: In the main algorithm, the step length l is fixed for ease of explanation. However, if the range [R_min, R_max] is very wide, it is also possible to define a set of intermediate encoding rates with variable step length. The rationale behind this modification is that the difference between 400 kbps and 600 kbps as target bitrate is much more significant than the difference between 4400 kbps and 4600 kbps. For example, one could define steps of length 200 Kbps under 1 Mbps, then steps of length 300 Kbps between 1 Mbps and 2 Mbps; 400 Kbps between 2 Mbps and 3 Mbps, and so on.

6.3. Varying frame rate and resolution

The trace-driven synthetic codec model explained in this section is relatively simple due to the choice of fixed frame rate and frame resolution. The model can be extended further to accommodate variable frame rate and/or variable spatial resolution.

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

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

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

7. Combining The Two Models

It is worthwhile noting that the statistical and trace-driven models each have their own advantages and drawbacks. Both models are fairly simple to implement. It takes significantly greater effort to fit the parameters of a statistical model to actual encoder output data. In contrast, it is straightforward for a trace-driven model to obtain encoded frame size data. Once validated, the statistical model is more flexible in mimicking a wide range of encoder/content behaviors by simply varying the corresponding parameters in the model. In this regard, a trace-driven model relies -- by definition -- on additional data collection efforts for accommodating new codecs or video contents.

In general, the trace-driven model is more realistic for mimicking the ongoing, steady-state behavior of a video traffic source with fluctuations around a constant target rate. In contrast, the statistical model is more versatile for simulating the behavior of a video stream in transient, such as when encountering sudden rate changes. It is also possible to combine both methods into a hybrid model. In this case, the steady-state behavior is driven by traces during steady state and the transient-state behavior is driven by the statistical model.

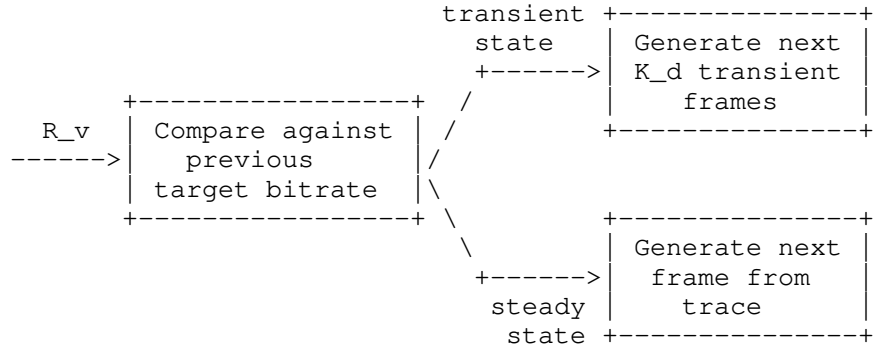

Figure 3: A hybrid video traffic model

As shown in Figure 3, the video traffic model operates in a transient state if the requested target rate R_v is substantially different from the previous target, or else it operates in steady state. During the transient state, a total of K_d frames are generated by the statistical model, resulting in one (1) big burst frame with size K_B followed by K_d-1 smaller frames. When operating at steady state, the video traffic model simply generates a frame according to the trace-driven model given the target rate, while modulating the frame interval according to the distribution specified by the statistical model. One example criterion for determining whether the traffic model should operate in a transient state is whether the rate change exceeds 10% of the previous target rate. Finally, as this model follows transient-state behavior dictated by the statistical model, upon a substantial rate change, the model will follow the time-damping mechanism as defined in Section 5.1, which is governed by parameter tau_v.

8. Implementation Status

The statistical, trace-driven, and hybrid models as described in this draft have been implemented as a stand-alone, platform-independent synthetic traffic source module. It can be easily integrated into network simulation platforms such as [ns-2] and [ns-3], as well as testbeds using a real network. The stand-alone traffic source module is available as an open source implementation at [Syncodecs].

9. IANA Considerations

There are no IANA impacts in this memo.

10. Security Considerations

The synthetic video traffic models as described in this draft do not impose any security threats. They are designed to mimic realistic traffic patterns for evaluating candidate RTP-based congestion control algorithms, so as to ensure stable operations of the network. It is RECOMMENDED that candidate algorithms be tested using the video traffic models presented in this draft before wide deployment over the Internet. If the generated synthetic traffic flows are sent over the Internet, they also need to be congestion controlled.

11. References

11.1. Normative References

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

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

11.2. Informative References

[H264] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services", May 2003, <https://www.itu.int/rec/T-REC-H.264>.

[HEVC] ITU-T Recommendation H.265, "High efficiency video coding", April 2013, <https://www.itu.int/rec/T-REC-H.265>.

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

[IETF-Interim] Zhu, X., Mena, S., and Z. Sarker, "Update on RMCAT Video Traffic Model: Trace Analysis and Model Update", April 2017, <https://www.ietf.org/proceedings/interim-2017-rmcat-01/slides/slides-interim-2017-rmcat-01-sessa-update-on-video-traffic-model-draft-00.pdf>.

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

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

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

[Papoulis] Papoulis, A., "Probability, Random Variables and Stochastic Processes", 2002.

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

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

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

Authors' Addresses

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

Email: xiaoqzhu@cisco.com

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

Email: semena@cisco.com

Zaheduzzaman Sarker
Ericsson AB
Luleae, SE 977 53
Sweden

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