RTP Media Congestion Avoidance Techniques
Internet-Draft
Intended status: Experimental
Expires: September 4, 2015

D. Hayes, Ed.
University of Oslo
S. Ferlin
Simula Research Laboratory
M. Welzl
University of Oslo
March 3, 2015

Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.
draft-hayes-rmcat-sbd-02

Abstract

This document describes a mechanism to detect whether end-to-end data flows share a common bottleneck. It relies on summary statistics that are calculated by a data receiver based on continuous measurements and regularly fed to a grouping algorithm that runs wherever the knowledge is needed. This mechanism complements the coupled congestion control mechanism in draft-welzl-rmcat-coupled-cc.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

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

1.1. The signals

The current Internet is unable to explicitly inform endpoints as to which flows share bottlenecks, so endpoints need to infer this from whatever information is available to them. The mechanism described here currently utilises packet loss and packet delay, but is not restricted to these.

1.1.1. Packet Loss

Packet loss is often a relatively rare signal. Therefore, on its own it is of limited use for SBD, however, it is a valuable supplementary measure when it is more prevalent.

1.1.2. Packet Delay

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

Measuring absolute OWD is difficult since it requires both the sender and receiver clocks to be synchronised. However, since the statistics being collected are relative to the mean OWD, a relative OWD measurement is sufficient. Clock drift is not usually significant over the time intervals used by this SBD mechanism (see [RFC6817] A.2 for a discussion on clock drift and OWD measurements). However, in circumstances where it is significant, Section 3.3.2 outlines a way of adjusting the calculations to cater for it.

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

1.1.3. Path Lag

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

2. Definitions

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

Acronyms used in this document:

OWD -- One Way Delay

PDV -- Packet Delay Variation

RTT -- Round Trip Time

SBD -- Shared Bottleneck Detection

Conventions used in this document:

T	--	the base time interval over which measurements are made.
N	--	the number of base time, T, intervals used in some calculations.
sum_T(...)	--	summation of all the measurements of the variable in parentheses taken over the interval T
sum(...)	--	summation of terms of the variable in parentheses
sum_N(...)	--	summation of N terms of the variable in parentheses

sum_NT(...) -- summation of all measurements taken over the interval N*T

E_T(...) -- the expectation or mean of the measurements of the variable in parentheses over T

E_N(...) -- The expectation or mean of the last N values of the variable in parentheses

E_M(...) -- The expectation or mean of the last M values of the variable in parentheses, where M <= N.

max_T(...) -- the maximum recorded measurement of the variable in parentheses taken over the interval T

min_T(...) -- the minimum recorded measurement of the variable in parentheses taken over the interval T

num_T(...) -- the count of measurements of the variable in parentheses taken in the interval T

num_VM(...) -- the count of valid values of the variable in parentheses given M records

PC -- a boolean variable indicating the particular flow was identified as experiencing congestion in the previous interval T (i.e. Previously Congested)

CD_T -- an estimate of the effect of Clock Drift on the mean OWD per T

CD_Adj(...) -- Mean OWD adjusted for clock drift

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

N, M, and F -- number of values (calculated over T).

2.1. Parameter Values

Reference [Hayes-LCN14] uses T=350ms, N=50, p_l = 0.1. The other parameters have been tightened to reflect minor enhancements to the algorithm outlined in Section 3.3: c_s = -0.01, p_f = p_s = p_d = 0.1, p_pdv = 0.2, p_v = 0.2. M=50, F=10, and c_h = 0.3 are additional parameters defined in the document. These are values that seem to work well over a wide range of practical Internet conditions, but are the subject of ongoing tests.

3. Mechanism

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

variance (estimate var_est, see Section 3.1.3)

skewness (estimate skew_est, see Section 3.1.2)

oscillation (estimate freq_est, see Section 3.1.4)

with packet loss (estimate pkt_loss, see Section 3.1.5) used as a supplementary statistic.

Summary statistics help to address both the noise and the path lag problems by describing the general shape over a relatively long period of time. This is sufficient for their application in coupled congestion control for RTP Media. They can be signalled from a receiver, which measures the OWD and calculates the summary statistics, to a sender, which is the entity that is transmitting the media stream. An RTP Media device may be both a sender and a receiver. SBD can be performed at either Sender or receiver or both.

```
                  +----+
                  | H2 |
                  +----+
                     |
                     | L2
                     |
         +----+  L1  |  L3  +----+
         | H1 |------|------| H3 |
         +----+             +----+
```

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

Figure 1

In Figure 1, there are two possible cases for shared bottleneck detection: a sender-based and a receiver-based case.

1. Sender-based: consider a situation where host H1 sends media streams to hosts H2 and H3, and L1 is a shared bottleneck. H2 and H3 measure the OWD and calculate summary statistics, which they send to H1 every T. H1, having this knowledge, can determine the shared bottleneck and accordingly control the send rates.

2. Receiver-based: consider that H2 is also sending media to H3, and L3 is a shared bottleneck. If H3 sends summary statistics to H1 and H2, neither H1 nor H2 alone obtain enough knowledge to detect this shared bottleneck; H3 can however determine it by combining the summary statistics related to H1 and H2, respectively. This case is applicable when send rates are controlled by the receiver; then, the signal from H3 to the senders contains the sending rate.

A discussion of the required signalling for the receiver-based case is beyond the scope of this document. For the sender-based case, the messages and their data format will be defined here in future versions of this document. We envision that an initialization message from the sender to the receiver could specify which key metrics are requested out of a possibly extensible set (pkt_loss, var_est, skew_est, freq_est). The grouping algorithm described in this document requires all four of these metrics, and receivers MUST be able to provide them, but future algorithms may be able to exploit other metrics (e.g. metrics based on explicit network signals). Moreover, the initialization message could specify T, N, and the necessary resolution and precision (number of bits per field).

3.1. Key metrics and their calculation

Measurements are calculated over a base interval, T. T should be long enough to provide enough samples for a good estimate of skewness, but short enough so that a measure of the oscillation can be made from N of these estimates. Reference [Hayes-LCN14] uses T = 350ms and N=M=50, which are values that seem to work well over a wide range of practical Internet conditions.

3.1.1. Mean delay

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

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

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

where M <= N. Generally M=N, setting M to be less than N allows the mechanism to be more responsive to changes, but potentially at the expense of a higher error rate (see Section 3.4 for a discussion on improving the responsiveness of the mechanism.)

3.1.2. Skewness Estimate

Skewness is difficult to calculate efficiently and accurately. Ideally it should be calculated over the entire period (M * T) from the mean OWD over that period. However this would require storing every delay measurement over the period. Instead, an estimate is made over T using the previous calculation of mean_delay. Comparisons are made using the mean of M skew estimates (an alternative that removes bias in the mean is given in Section 3.3.3).

The skewness is estimated using two counters, counting the number of one way delay samples (OWD) above and below the mean:

```
skew_est_T =  (sum_T(OWD < mean_delay)

              - sum_T(OWD > mean_delay)) / num_T(OWD)

   where

      if (OWD < mean_delay) 1 else 0

      if (OWD > mean_delay) 1 else 0

   skew_est_T is a number between -1 and 1

skew_est = E_M(skew_est_T) = sum_M(skew_est_T) / M
```

For implementation ease, mean_delay does not include the mean of the current T interval.

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

3.1.3. Variance Estimate

Packet Delay Variation (PDV) ([RFC5481] and [ITU-Y1540]) is used as an estimator of the variance of the delay signal. We define PDV as follows:

```
PDV = PDV_max = max_T(OWD) - E_T(OWD)

var_est = E_M(PDV) = sum_M(PDV) / M
```

This modifies PDV as outlined in [RFC5481] to provide a summary statistic version that best aids the grouping decisions of the algorithm (see [Hayes-LCN14] section IVB).

The use of PDV = PDV_min = E_T(OWD) - min_T(OWD) is currently being investigated as an alternative that is less sensitive to noise. The drawback of using PDV_min is that it does not distinguish between groups of flows with similar values of skew_est as well as PDV_max (see [Hayes-LCN14] section IVB).

3.1.4. Oscillation Estimate

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

```
freq_est = number_of_crossings / N
```

Where

we define a significant mean crossing as a crossing that extends p_v * var_est from mean_delay. In our experiments we have found that p_v = 0.2 is a good value.

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

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

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

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

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

3.1.5. Packet loss

The proportion of packets lost is used as a supplementary measure:

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

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

3.2. Flow Grouping

3.2.1. Flow Grouping Algorithm

The following grouping algorithm is RECOMMENDED for SBD in the RMCAT context and is sufficient and efficient for small to moderate numbers of flows. For very large numbers of flows (e.g. hundreds), a more complex clustering algorithm may be substituted.

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

Flows determined to be experiencing congestion are successively divided into groups based on freq_est, var_est, and skew_est.

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

1. Grouping will be performed on flows where:

   ```
   skew_est < c_s

      || ( skew_est < c_h && PC )

      || pkt_loss > p_l
   ```

   The parameter c_s controls how sensitive the mechanism is in detecting congestion. C_s = 0.0 was used in [Hayes-LCN14]. A value of c_s = 0.05 is a little more sensitive, and c_s = -0.05 is a little less sensitive. C_h controls the hysteresis on flows that were grouped as experiencing congestion last time.

   These flows, flows experiencing congestion, are then progressively divided into groups based on the freq_est, PDV, and skew_est summary statistics. The process proceeds according to the following steps:

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

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

3. Group flows whose difference in sorted E_N(PDV) (highest to lowest) is less than a threshold:

   ```
   diff(var_est) < (p_pdv * var_est)
   ```

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

4. Group flows whose difference in sorted skew_est or pkt_loss is less than a threshold:

   ```
   if pkt_loss < p_l

      diff(skew_est) < p_s

   otherwise

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

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

   This procedure involves sorting estimates from highest to lowest. It is simple to implement, and efficient for small numbers of flows, such as are expected in RTCWEB.

3.2.2. Using the flow group signal

A grouping decisions is made every T from the second T, though they will not attain their full design accuracy until after the N'th T interval.

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

3.3. Removing Noise from the Estimates

The following describe small changes to the calculation of the key metrics that help remove noise from them. Currently these "tweaks" are described separately to keep the main description succinct. In future revisions of the draft these enhancements may replace the original key metric calculations.

3.3.1. Oscillation noise

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

To remove this source of noise from freq_est:

1. Set the current PDV to PDV = NaN (a value representing an invalid record, ie Not a Number) for flows that are deemed to not be experiencing congestion by the first skew_est based grouping test (see Section 3.2.1).

2. Then var_est = sum_M(PDV != NaN) / num_VM(PDV)

3. For freq_est, only record a significant mean crossing if flow is experiencing congestion.

These three changes will remove the non-congestion noise from freq_est.

3.3.2. Clock drift

Generally sender and receiver clock drift will be too small to cause significant errors in the estimators. Skew_est is most sensitive to this type of noise. In circumstances where clock drift is high, making $M < N$ can reduce this error.

A better method is to estimate the effect the clock drift is having on the E_N(E_T(OWD)), and then adjust mean_delay accordingly. A simple method of doing this follows:

First divide the N E_T(OWD) values into two halves (N/2 in each) -- old and new.

Calculate a mean of the old half:

```
Older_mean = E_old(E_T(OWD)) / N/2
```

Calculate a mean of the new (most recent) half:

```
Newer_mean = E_new(E_T(OWD)) / N/2
```

A linear estimate of the Clock Drift per T estimates is:

```
CD_T = (Newer_mean - Older_mean)/N/2
```

An adjusted mean estimate then is:

```
mean_delay = CD_Adj(E_M(E_T(OWD))) = E_M(E_T(OWD)) + CD_T * M/2
```

CD_Adj can be thought of as a prediction of what the long term mean will be in the current measurement period T. It is used as the basis for skew_est and freq_est.

3.3.3. Bias in the skewness measure

If successive calculations of skew_est are made with very different numbers of samples (num_T(OWD)), the simple calculation of E_M(skew_est) used for grouping decisions will be biased by the intervals that have few samples samples. This bias can be corrected if necessary as follows.

```
    skew_base_T = (sum_T(OWD < mean_delay

                  - sum_T(OWD > mean_delay)

       skew_est = sum_MT(skew_base_T)/num_MT(OWD)
```

This calculation requires slightly more state, since an implementation will need to maintain two cyclic buffers storing skew_base_T and num_T(OWD) respectively to manage the rolling summations (note only one cyclic buffer is needed for the calculation of skew_est outlined previously).

3.4. Reducing lag and Improving Responsiveness

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

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

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

Although more recent measurements are more valuable, older measurements are still needed to gain an accurate estimate of the distribution descriptor we are measuring. Unfortunately, the simple exponentially weighted moving average weights drop off too quickly for our requirements and have an infinite tail. A simple linearly declining weighted moving average also does not provide enough weight to the most recent measurements. We propose a piecewise linear distribution of weights, such that the first section (samples 1:F) is flat as in a simple moving average, and the second section (samples F+1:M) is linearly declining weights to the end of the averaging window. We choose integer weights, which allows incremental calculation without introducing rounding errors.

3.4.1. Improving the response of the skewness estimate

The weighted moving average for skew_est, based on skew_est in Section 3.3.3, can be calculated as follows:

```
      skew_est = ((M-F+1)*sum(skew_base_T(1:F))

                      + sum([(M-F):1].*skew_base_T(F+1:M)))

                 / ((M-F+1)*sum(numsampT(1:F))

                      + sum([(M-F):1].*numsampT(F+1:M)))
```

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

3.4.2. Improving the response of the variance estimate

The weighted moving average for var_est can be calculated as follows:

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

                / (F*(M-F+1) + sum([(M-F):1])
```

where 1:F refers to the integer values 1 through to F, and [(M-F):1] refers to an array of the integer values (M-F) declining through to 1; and ".*" is the array scalar dot product operator. When removing oscillation noise (see Section 3.3.1) this calculation must be adjusted to allow for invalid PDV records.

4. Measuring OWD

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

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

4.1. Time stamp resolution

The SBD mechanism requires timing information precise enough to be able to make comparisons. As a rule of thumb, the time resolution should be less than one hundredth of a typical path’s range of delays. In general, the lower the time resolution, the more care that needs to be taken to ensure rounding errors do not bias the skewness calculation.

Typical RTP media flows use sub-millisecond timers, which should be adequate in most situations.

5. Acknowledgements

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

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

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

Non-authenticated RTCP packets carrying shared bottleneck indications and summary statistics could allow attackers to alter the bottleneck sharing characteristics for private gain or disruption of other parties communication.

8. Change history

Changes made to this document:

01->02 :	New section describing improvements to the key metric calculations that help to remove noise, bias, and reduce lag. Some revisions to the notation to make it clearer. Some tightening of the thresholds.
00->01 :	Revisions to terminology for clarity

9. References

9.1. Normative References

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

9.2. Informative References

[Hayes-LCN14]
Hayes, D., Ferlin, S., and M. Welzl, "Practical Passive Shared Bottleneck Detection using Shape Summary Statistics", Proc. the IEEE Local Computer Networks (LCN) p150-158, September 2014, <http://heim.ifi.uio.no/davihay/hayes14__pract_passiv_shared_bottl_detec-abstract.html>.

[I-D.welzl-rmcat-coupled-cc]
Welzl, M., Islam, S., and S. Gjessing, "Coupled congestion control for RTP media", draft-welzl-rmcat-coupled-cc-04 (work in progress), October 2014.

[ITU-Y1540]
ITU-T, "Internet Protocol Data Communication Service - IP Packet Transfer and Availability Performance Parameters", Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks , March 2011, <http://www.itu.int/rec/T-REC-Y.1540-201103-I/en>.

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

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control

Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, March 2009.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, December 2012.

Authors’ Addresses

David Hayes (editor)
University of Oslo
PO Box 1080 Blindern
Oslo,   N-0316
Norway

Phone: +47 2284 5566
Email: davihay@ifi.uio.no

Simone Ferlin
Simula Research Laboratory
P.O.Box 134
Lysaker,   1325
Norway

Phone: +47 4072 0702
Email: ferlin@simula.no

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

Phone: +47 2285 2420
Email: michawe@ifi.uio.no