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

D. Hayes, Ed.
University of Oslo
S. Ferlin
Simula Research Laboratory
M. Welzl
K. Hiorth
University of Oslo

March 21, 2016

Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.
draft-ietf-rmcat-sbd-04

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

1.1. The signals

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

1.1.1. Packet Loss

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

1.1.2. Packet Delay

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

Measuring absolute OWD is difficult since it requires both the sender and receiver clocks to be synchronised. However, since the statistics being collected are relative to the mean OWD, a relative

OWD measurement is sufficient. Clock skew is not usually significant over the time intervals used by this SBD mechanism (see [RFC6817] A.2 for a discussion on clock skew and OWD measurements). However, in circumstances where it is significant, Section 3.4.2 outlines a way of adjusting the calculations to cater for it.

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

1.1.3. Path Lag

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

2. Definitions

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

Acronyms used in this document:

OWD -- One Way Delay

MAD -- Mean Absolute Deviation

RTT -- Round Trip Time

SBD -- Shared Bottleneck Detection

Conventions used in this document:

T -- the base time interval over which measurements are made.

N -- the number of base time, T, intervals used in some calculations.

M -- the number of base time, T, intervals used in some calculations.

```
      sum_T(...) --  summation of all the measurements of the variable
                     in parentheses taken over the interval T

      sum(...)   --  summation of terms of the variable in parentheses

      sum_N(...) --  summation of N terms of the variable in parentheses

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

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

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

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

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

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

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

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

      PB --          a boolean variable indicating the particular flow
                     was identified transiting a bottleneck in the
                     previous interval T (i.e.  Previously Bottleneck)

      skew_est --    a measure of skewness in a OWD distribution.

     skew_base_T --  a variable used as an intermediate step in
                     calculating skew_est.

      var_est --     a measure of variability in OWD measurements.

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

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

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

M and F -- number of values related to N

.

2.1. Parameters and their Effect

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

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

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

c_s — c_s is the threshold in skew_est used for determining whether a flow is transiting a bottleneck or not. It should be slightly negative so that a very lightly loaded path does not give a false indication. Setting c_s more negative makes the SBD mechanism less sensitive to transient and slight bottlenecks.

c_h — c_h adds hysteresis to the botteneck determination. It should be large enough to avoid constant switching in the determination, but low enough to ensure that grouping is not attempted when there is no bottleneck and the delay and loss signals cannot be relied upon.

p_v — p_v determines the sensitivity of freq_est to noise. Making it smaller will yield higher but noisier values for freq_est. Making it too large will render it ineffective for determining groups.

p_* — Flows are separated when the skew_est|var_est|freq_est measure is greater than p_s|p_f|p_d|p_mad. Adjusting these is a compromise between false grouping of flows that do not share a bottleneck and false splitting of flows that do. Making them larger can help if the measures are very noisy, but reducing the noise in the statistical measures by adjusting T and N|M may be a better solution.

2.2. Recommended Parameter Values

Reference [Hayes-LCN14] uses T=350ms, N=50, p_l=0.1. The other parameters have been tightened to reflect minor enhancements to the algorithm outlined in Section 3.4: c_s=-0.01, p_f=p_d=0.1, p_s=0.15, p_mad=0.1, p_v=0.7. M=30, F=20, and c_h = 0.3 are additional parameters defined in the document. These are values that seem to work well over a wide range of practical Internet conditions.

3. Mechanism

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

variability (estimate var_est, see Section 3.2.3)

skewness (estimate skew_est, see Section 3.2.2)

oscillation (estimate freq_est, see Section 3.2.4)

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

Summary statistics help to address both the noise and the path lag problems by describing the general shape over a relatively long period of time. Each summary statistic portrays a "view" of the bottleneck link characteristics, and when used together, they provide a robust discrimination for grouping flows. They can be signalled from a receiver, which measures the OWD and calculates the summary statistics, to a sender, which is the entity that is transmitting the media stream. An RTP Media device may be both a sender and a receiver. SBD can be performed at either a sender or a receiver or both.

```
                         +----+
                         | 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 locations for shared bottleneck detection: sender-side and receiver-side.

1. Sender-side: consider a situation where host H1 sends media streams to hosts H2 and H3, and L1 is a shared bottleneck. H2 and H3 measure the OWD and packet loss and either send back this raw data, or the calculated summary statistics, periodically to H1 every T. H1, having this knowledge, can determine the shared bottleneck and accordingly control the send rates.

2. Receiver-side: consider that H2 is also sending media to H3, and L3 is a shared bottleneck. If H3 sends summary statistics to H1 and H2, neither H1 nor H2 alone obtain enough knowledge to detect this shared bottleneck; H3 can however determine it by combining the summary statistics related to H1 and H2, respectively.

3.1. SBD feedback requirements

There are three possible scenarios each with different feedback requirements:

1. Both summary statistic calculations and SBD are performed at senders only.

2. Summary statistics calculated on the receivers and SBD at the senders.

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

3.1.1. Feedback when all the logic is placed at the sender

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

The sender requires precise accurate OWD measurements for every packet, along with the proportion of packets lost over the interval T, to be sent from the receivers to the senders every T.

An initialisation message may be required to agree on the feedback interval.

3.1.2. Feedback when the statistics are calculated at the receiver and SBD at the sender

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

- An initialization message from the sender to the receiver will contain the following information:

  * A protocol identifier (SBD=01). This is to future proof the message exchange so that potential advances in SBD technology can be easily deployed. All following initialisation elements relate to the mechanism outlined in this document which will have the identifier SBD=01.

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

  * The values of T, N, M, and the necessary resolution and precision of the relayed statistics.

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

This initialisation exchange may be repeated to finalize the agreed metrics should not all be supported by all receivers.

After initialisation the agreed summary statistics will be fed back to the sender every T.

3.1.3. Feedback when bottlenecks can be determined at both senders and receivers

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

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

After initialisation both summary statistics and shared bottleneck determinations will need to be exchanged every T.

3.2. Key metrics and their calculation

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

3.2.1. Mean delay

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

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

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

where M <= N. Setting M to be less than N allows the mechanism to be more responsive to changes, but potentially at the expense of a higher error rate (see Section 3.5 for a discussion on improving the responsiveness of the mechanism.)

3.2.2. Skewness Estimate

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

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

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

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

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

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

   where skew_est is a number between -1 and 1
```

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

3.2.3. Variability Estimate

Mean Absolute Deviation (MAD) delay is a robust variability measure that copes well with different send rates. It can be implemented in an online manner as follows:

```
var_base_T = sum_T(|OWD - E_T(OWD)|)

   where

      |x| is the absolute value of x

      E_T(OWD) is the mean OWD calculated in the previous T

var_est = MAD_MT = sum_MT(var_base_T)/num_MT(OWD)
```

For calculation of freq_est p_v=0.7

For the grouping threshold p_mad=0.1

3.2.4. Oscillation Estimate

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

freq_est = number_of_crossings / N

where we define a significant mean crossing as a crossing that extends p_v * var_est from mean_delay. In our experiments we have found that p_v = 0.7 is a good value.

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

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

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

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

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

3.2.5. Packet loss

The proportion of packets lost over the period NT is used as a supplementary measure:

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

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

3.3. Flow Grouping

3.3.1. Flow Grouping Algorithm

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

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

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

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

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

   ```
   skew_est < c_s

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

The parameter c_s controls how sensitive the mechanism is in detecting a bottleneck. C_s = 0.0 was used in [Hayes-LCN14]. A value of c_s = 0.05 is a little more sensitive, and c_s = -0.05 is a little less sensitive. C_h controls the hysteresis on flows that were grouped as transiting a bottleneck last time. If the test result is TRUE, PB=TRUE, otherwise PB=FALSE.

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

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

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

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

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

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

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

   diff(skew_est) < p_s

5. When packet loss is high enough to be reliable (pkt_loss > p_l), group flows whose difference is less than a threshold

   diff(pkt_loss) < (p_d * pkt_loss)

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

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

3.3.2. Using the flow group signal

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

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

3.4. Removing Noise from the Estimates

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

3.4.1. Oscillation noise

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

To remove this source of noise from freq_est:

1. Set the current var_base_T = NaN (a value representing an invalid record, i.e. Not a Number) for flows that are deemed to not be transiting a bottleneck by the first skew_est based grouping test (see Section 3.3.1).

2. Then var_est = sum_MT(var_base_T != NaN) / num_MT(OWD)

3. For freq_est, only record a significant mean crossing if flow deemed to be transiting a bottleneck.

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

3.4.2. Clock skew

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

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

3.5. Reducing lag and Improving Responsiveness

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

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

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

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

3.5.1. Improving the response of the skewness estimate

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

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

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

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

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

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

To calculate this weighted skew_est incrementally:

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

Initialise: sum_skewbase = 0, F_skewbase=0, W_D_skewbase=0

skewbase_hist = buffer length M initialize to 0

numsampT = buffer length M initialzed to 0

Steps per iteration:

1. old_skewbase = skewbase_hist(M)

2. old_numsampT = numsampT(M)

3. cycle(skewbase_hist)

4. cycle(numsampT)

5. numsampT(1) = num_T(OWD)

6. skewbase_hist(1) = skew_base_T

7. F_skewbase = F_skewbase + skew_base_T - skewbase_hist(F+1)

8. W_D_skewbase = W_D_skewbase + (M-F)*skewbase_hist(F+1) - sum_skewbase

9. W_D_numsamp = W_D_numsamp + (M-F)*numsampT(F+1) - sum_numsamp + F_numsamp

10. F_numsamp = F_numsamp + numsampT(1) - numsampT(F+1)

11. sum_skewbase = sum_skewbase + skewbase_hist(F+1) - old_skewbase

12. sum_numsamp = sum_numsamp + numsampT(1) - old_numsampT

13. skew_est = ((M-F+1)*F_skewbase + W_D_skewbase) / ((M-F+1)*F_numsamp+W_D_numsamp)

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

3.5.2. Improving the response of the variability estimate

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

```
var_est =  ((M-F+1)*sum(var_base_T(1:F))

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

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

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

where numsampT is an array of the number of OWD samples in each T (i.e. num_T(OWD)), and numsampT(1) is the most recent; skew_base_T(1) is the most recent calculation of skew_base_T; 1:F refers to the integer values 1 through to F, and [(M-F):1] refers to an array of the integer values (M-F) declining through to 1; and ".*" is the array scalar dot product operator. When removing oscillation noise (see Section 3.4.1) this calculation must be adjusted to allow for invalid var_base_T records.

Var_est can be calculated incrementally in the same way as skew_est in Section 3.5.1. However, note that the buffer numsampT is used for both calculations so the operations on it should not be repeated.

4. Measuring OWD

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

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

4.1. Time stamp resolution

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

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

5. Implementation status

The University of Oslo is currently working on an implementation of this in the Chromium browser.

6. Acknowledgements

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

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

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

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

9. Change history

Changes made to this document:

WG-03->WG-04 : Add M to terminology table, suggest skew_est based on previous T and no freq_est in clock skew section, feedback requirements as a separate sub section.

WG-02->WG-03 : Correct misspelled author

WG-01->WG-02 : Removed ambiguity associated with the term "congestion". Expanded the description of initialisation messages. Removed PDV metric. Added description of incremental weighted metric calculations for skew_est. Various clarifications based on implementation work. Fixed typos and tuned parameters.

WG-00->WG-01 : Moved unbiased skew section to replace skew estimate, more robust variability estimator, the term variance replaced with variability, clock drift term corrected to clock skew, revision to clock skew section with a place holder, description of parameters.

02->WG-00 : Fixed missing 0.5 in 3.3.2 and missing brace in 3.3.3

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

00->01 : Revisions to terminology for clarity

10. References

10.1. Normative References

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

10.2. Informative References

[Hayes-LCN14]
Hayes, D., Ferlin, S., and M. Welzl, "Practical Passive Shared Bottleneck Detection using Shape Summary Statistics", Proc. the IEEE Local Computer Networks (LCN) pp150-158, September 2014, <http://heim.ifi.uio.no/davihay/hayes14__pract_passiv_shared_bottl_detec-abstract.html>.

[I-D.ietf-rmcat-coupled-cc]
Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-00 (work in progress), September 2015.

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

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

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <http://www.rfc-editor.org/info/rfc5124>.

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

[Zhang-Infocom02] Zhang, L., Liu, Z., and H. Xia, "Clock synchronization algorithms for network measurements", Proc. the IEEE International Conference on Computer Communications (INFOCOM) pp160-169, September 2002, <http://dx.doi.org/10.1109/INFCOM.2002.1019257>.

Authors' Addresses

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

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

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

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

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

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

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

Email: kristahi@ifi.uio.no