Network Working Group   S. Holmer
Internet-Draft   H. Lundin
Intended status: Informational   Google
Expires: December 31, 2015   G. Carlucci
                                                             L. De Cicco
                                                               S. Mascolo
                                                       Politecnico di Bari
                                                            June 29, 2015

       A Google Congestion Control Algorithm for Real-Time Communication
                    draft-alvestrand-rmcat-congestion-03

Abstract

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

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

Requirements Language

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

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

1.1. Mathematical notation conventions

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

    The following notational conventions are used:

    X_bar The variable X, where X is a vector - conventionally marked by
       a bar on top of the variable name.

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

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

    [x y z] A row vector consisting of elements x, y and z.

   X_bar^T The transpose of vector X_bar.

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

2. System model

   The following elements are in the system:

   o RTP packet - an RTP packet containing media data.

   o Packet group - a set of RTP packets transmitted from the sender
      uniquely identified by the group departure and group arrival time
      (absolute send time) [abs-send-time]. These could be video
      packets, audio packets, or a mix of audio and video packets.

   o Incoming media stream - a stream of frames consisting of RTP
      packets.

   o RTP sender - sends the RTP stream over the network to the RTP
      receiver. It generates the RTP timestamp and the abs-send-time
      header extension

   o RTP receiver - receives the RTP stream, marks the time of arrival.

   o RTCP sender at RTP receiver - sends receiver reports, REMB
      messages and transport-wide RTCP feedback messages.

   o RTCP receiver at RTP sender - receives receiver reports and REMB
      messages and transport-wide RTCP feedback messages, reports these
      to the sender side controller.

   o RTCP receiver at RTP receiver, receives sender reports from the
      sender.

   o Loss-based controller - takes loss rate measurement, round trip
      time measurement and REMB messages, and computes a target sending
      bitrate.

   o Delay-based controller - takes the packet arrival info, either at
      the RTP receiver, or from the feedback received by the RTP sender,
      and computes a maximum bitrate which it passes to the loss-based
      controller.

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

3. Feedback and extensions

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

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

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

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

4. Delay-based control

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

4.1. Arrival-time model

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

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

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

At the receiving side we are observing groups of incoming packets, where a group of packets is defined as follows:

o A sequence of packets which are sent within a burst_time interval
   constitute a group. RECOMMENDED value for burst_time is 5 ms.

o In addition, any packet which has an inter-arrival time less than
   burst_time and an inter-group delay variation d(i) less than 0 is
   also considered being part of the current group of packets. The
   reasoning behind including these packets in the group is to better
   handle delay transients, caused by packets being queued up for
   reasons unrelated to congestion. As an example this has been
   observed to happen on many Wi-Fi and wireless networks.

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

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

Since the time ts to send a group of packets of size L over a path with a capacity of C is roughly

    ts = L/C

we can model the inter-group delay variation as:

```
   d(i) = L(i)/C(i) - L(i-1)/C(i-1) + w(i) =

        L(i)-L(i-1)
     = ------------- + w(i) = dL(i)/C(i) + w(i)
            C(i)
```

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

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

Equation 1

```
   d(i) = dL(i)/C(i) + m(i) + v(i)
```

This is our fundamental model, where we take into account that a large group of packets need more time to traverse the link than a small group, thus arriving with higher relative delay. The noise term represents network jitter and other delay effects not captured by the model.

4.2. Arrival-time filter

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

Let

```
   theta_bar(i) = [1/C(i)  m(i)]^T
```

and call it the state at time i. We model the state evolution from time i to time i+1 as

```
   theta_bar(i+1) = theta_bar(i) + u_bar(i)
```

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

```
   Q(i) = E{u_bar(i) * u_bar(i)^T}
```

Q(i) is RECOMMENDED as a diagonal matrix with main diagonal elements
as:

```
   diag(Q(i)) = [10^-13 10^-3]^T
```

Given equation 1 we get

```
   d(i) = h_bar(i)^T * theta_bar(i) + v(i)
```

```
   h_bar(i) = [dL(i)  1]^T
```

where v(i) is zero mean white Gaussian measurement noise with
variance var_v = sigma(v,i)^2

The Kalman filter recursively updates our estimate

```
   theta_hat(i) = [1/C_hat(i) m_hat(i)]^T
```

as

```
   z(i) = d(i) - h_bar(i)^T * theta_hat(i-1)
```

```
   theta_hat(i) = theta_hat(i-1) + z(i) * k_bar(i)
```

```
                   ( E(i-1) + Q(i) ) * h_bar(i)
   k_bar(i) = ---------------------------------------------------
             var_v_hat(i) + h_bar(i)^T * (E(i-1) + Q(i)) * h_bar(i)
```

```
   E(i) = (I - k_bar(i) * h_bar(i)^T) * (E(i-1) + Q(i))
```

where I is the 2-by-2 identity matrix.

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

```
   var_v_hat(i) = max(beta * var_v_hat(i-1) + (1-beta) * z(i)^2, 1)
```

```
   beta = (1-chi)^(30/(1000 * f_max))
```

where f_max = max {1/(T(j) - T(j-1))} for j in i-K+1,...,i is the
highest rate at which the last K packet groups have been received and
chi is a filter coefficient typically chosen as a number in the
interval [0.1, 0.001]. Since our assumption that v(i) should be zero
mean WGN is less accurate in some cases, we have introduced an
additional outlier filter around the updates of var_v_hat. If z(i) >
3*sqrt(var_v_hat) the filter is updated with 3*sqrt(var_v_hat) rather

than z(i). For instance v(i) will not be white in situations where
packets are sent at a higher rate than the channel capacity, in which
case they will be queued behind each other.

4.3. Over-use detector

The offset estimate m(i), obtained as the output of the arrival-time
filter, is compared with a threshold gamma_1(i). An estimate above
the threshold is considered as an indication of over-use. Such an
indication is not enough for the detector to signal over-use to the
rate control subsystem. A definitive over-use will be signaled only
if over-use has been detected for at least gamma_2 milliseconds.
However, if m(i) < m(i-1), over-use will not be signaled even if all
the above conditions are met. Similarly, the opposite state, under-
use, is detected when m(i) < -gamma_1(i). If neither over-use nor
under-use is detected, the detector will be in the normal state.

The threshold gamma_1 has a remarkable impact on the overall dynamics
and performance of the algorithm. In particular, it has been shown
that using a static threshold gamma_1, a flow controlled by the
proposed algorithm can be starved by a concurrent TCP flow [Pv13].
This starvation can be avoided by increasing the threshold gamma_1 to
a sufficiently large value.

The reason is that, by using a larger value of gamma_1, a larger
queuing delay can be tolerated, whereas with a small gamma_1, the
over-use detector quickly reacts to a small increase in the offset
estimate m(i) by generating an over-use signal that reduces the
delay-based estimate of the available bandwidth A_hat (see
Section 4.4). Thus, it is necessary to dynamically tune the
threshold gamma_1 to get good performance in the most common
scenarios, such as when competing with loss-based flows.

For this reason, we propose to vary the threshold gamma_1(i)
according to the following dynamic equation:

gamma_1(i) = gamma_1(i-1) + (t(i)-t(i-1)) * K(i) * (|m(i)|-gamma_1(i-1))

with K(i)=K_d if |m(i)| < gamma_1(i-1) or K(i)=K_u otherwise. The
rationale is to increase gamma_1(i) when m(i) is outside of the range
[-gamma_1(i-1),gamma_1(i-1)], whereas, when the offset estimate m(i)
falls back into the range, gamma_1 is decreased. In this way when
m(i) increases, for instance due to a TCP flow entering the same
bottleneck, gamma_1(i) increases and avoids the uncontrolled
generation of over-use signals which may lead to starvation of the
flow controlled by the proposed algorithm [Pv13]. Moreover,
gamma_1(i) SHOULD NOT be updated if this condition holds:

   $|m(i)|$ - gamma_1(i) > 15

   It is also RECOMMENDED to clamp gamma_1(i) to the range [6, 600],
   since a too small gamma_1(i) can cause the detector to become overly
   sensitive.

   On the other hand, when m(i) falls back into the range
   [-gamma_1(i-1),gamma_1(i-1)] the threshold gamma_1(i) is decreased so
   that a lower queuing delay can be achieved.

   It is RECOMMENDED to choose K_u > K_d so that the rate at which
   gamma_1 is increased is higher than the rate at which it is
   decreased. With this setting it is possible to increase the
   threshold in the case of a concurrent TCP flow and prevent starvation
   as well as enforcing intra-protocol fairness. RECOMMENDED values for
   gamma_1(0), gamma_2, K_u and K_d are respectively 12.5 ms, 10 ms,
   0.01 and 0.00018.

4.4. Rate control

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

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

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

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

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

```
+----+--------+-----------+-----------+--------+
|       \ State |   Hold    |  Increase |Decrease|
|         \     |           |           |        |
|  Signal\      |           |           |        |
+--------+----+-----------+-----------+--------+
|  Over-use  | Decrease  | Decrease  |        |
+-----------+-----------+-----------+--------+
|  Normal    | Increase  |           |  Hold  |
+-----------+-----------+-----------+--------+
|  Under-use |           |   Hold    |  Hold  |
+-----------+-----------+-----------+--------+
```

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

The system does a multiplicative increase if the current bandwidth
estimate appears to be far from convergence, while it does an
additive increase if it appears to be closer to convergence. We
assume that we are close to convergence if the currently incoming
bitrate, $R\_hat(i)$, is close to an average of the incoming bitrates at
the time when we previously have been in the Decrease state. "Close"
is defined as three standard deviations around this average. It is
RECOMMENDED to measure this average and standard deviation with an
exponential moving average with the smoothing factor 0.95, as it is
expected that this average covers multiple occasions at which we are
in the Decrease state. Whenever valid estimates of these statistics
are not available, we assume that we have not yet come close to
convergence and therefore remain in the multiplicative increase
state.

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

$R\_hat(i)$ is the incoming bitrate measured by the delay-based
controller over a T seconds window:

   $R\_hat(i) = 1/T * sum(L(j))$ for j from 1 to $N(i)$

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

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

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

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

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

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

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

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

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

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

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

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

When the detector signals under-use to the rate control subsystem, we know that queues in the network path are being emptied, indicating that our available bandwidth estimate A_hat is lower than the actual available bandwidth. Upon that signal the rate control subsystem will enter the hold state, where the receive-side available bandwidth

estimate will be held constant while waiting for the queues to
stabilize at a lower level - a way of keeping the delay as low as
possible. This decrease of delay is wanted, and expected,
immediately after the estimate has been reduced due to over-use, but
can also happen if the cross traffic over some links is reduced.

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

4.5. Parameters settings

Parameter	Description	RECOMMENDED Value
burst_time	Time limit in milliseconds between packet bursts which identifies a group	5 ms
Q	State noise covariance matrix	diag(Q(i)) = [10^-13 10^-3]^T
E(0)	Initial value of the system error covariance	diag(E(0)) = [100 0.1]^T
chi	Coefficient used for the measured noise variance	[0.1, 0.001]
gamma_1(0)	Initial value for the adaptive threshold	12.5 ms
gamma_2	Time required to trigger an overuse signal	10 ms
K_u	Coefficient for the adaptive threshold	0.01
K_d	Coefficient for the adaptive threshold	0.00018
T	Time window for measuring the received bitrate	[0.5, 1] s
alpha	Decrease rate factor	0.85

Table 1: RECOMMENDED values for delay based controller

Table 1

5. Loss-based control

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

estimates computed by the loss-based controller are denoted with
As_hat.

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

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

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

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

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

The new bandwidth estimate is lower-bounded by the TCP Friendly Rate
Control formula [RFC3448] and upper-bounded by the delay-based
estimate of the available bandwidth A_hat(i), where the delay-based
estimate has precedence:

$$As\_hat(i) >= \frac{8\ s}{R*sqrt(2*b*p/3) + (t\_RTO*(3*sqrt(3*b*p/8)*p*(1+32*p^2)))}$$

$$As\_hat(i) <= A\_hat(i)$$

where b is the number of packets acknowledged by a single TCP
acknowledgment (set to 1 per TFRC recommendations), t_RTO is the TCP
retransmission timeout value in seconds (set to 4*R) and s is the
average packet size in bytes. R is the round-trip time in seconds.

(The multiplication by 8 comes because TFRC is computing bandwidth in
bytes, while this document computes bandwidth in bits.)

In words: The loss-based estimate will never be larger than the
delay-based estimate, and will never be lower than the estimate from
the TFRC formula except if the delay-based estimate is lower than the
TFRC estimate.

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

6. Interoperability Considerations

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

7. Implementation Experience

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

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

8. Further Work

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

Work that can be done on this basis includes:

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

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

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

9. IANA Considerations

   This document makes no request of IANA.

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

10. Security Considerations

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

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

11. Acknowledgements

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

12. References

12.1. Normative References

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

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

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

   [RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP
              Friendly Rate Control (TFRC): Protocol Specification", RFC
              3448, January 2003.

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

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

12.2. Informative References

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

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

Appendix A. Change log

A.1. Version -00 to -01

   o Added change log

   o Added appendix outlining new extensions

   o Added a section on when to send feedback to the end of section 3.3
      "Rate control", and defined min/max FB intervals.

   o Added size of over-bandwidth estimate usage to "further work"
      section.

   o Added startup considerations to "further work" section.

   o Added sender-delay considerations to "further work" section.

   o Filled in acknowledgments section from mailing list discussion.

A.2. Version -01 to -02

   o Defined the term "frame", incorporating the transmission time
      offset into its definition, and removed references to "video
      frame".

   o Referred to "m(i)" from the text to make the derivation clearer.

   o Made it clearer that we modify our estimates of available
      bandwidth, and not the true available bandwidth.

   o Removed the appendixes outlining new extensions, added pointers to
      REMB draft and RFC 5450.

A.3. Version -02 to -03

   o Added a section on how to process multiple streams in a single
      estimator using RTP timestamps to NTP time conversion.

   o Stated in introduction that the draft is aimed at the RMCAT
      working group.

A.4. rtcweb-03 to rmcat-00

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

A.5. rmcat -00 to -01

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

A.6. rmcat -01 to -02

   o Added Luca De Cicco and Saverio Mascolo as authors.

   o Extended the "Over-use detector" section with new technical
      details on how to dynamically tune the offset gamma_1 for improved
      fairness properties.

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

A.7. rmcat -02 to -03

   o Swapped receiver-side/sender-side controller with delay-based/
      loss-based controller as there is no longer a requirement to run
      the delay-based controller on the receiver-side.

   o Removed the discussion about multiple streams and transmission
      time offsets.

   o Introduced a new section about "Feedback and extensions".

   o Improvements to the threshold adaptation in the "Over-use
      detector" section.

   o Swapped the previous MIMD rate control algorithm for a new AIMD
      rate control algorithm.

Authors' Addresses

   Stefan Holmer
   Google
   Kungsbron 2
   Stockholm 11122
   Sweden

   Email: holmer@google.com

   Henrik Lundin
   Google
   Kungsbron 2
   Stockholm 11122
   Sweden

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

   Email: gaetano.carlucci@poliba.it

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

   Email: l.decicco@poliba.it

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

   Email: mascolo@poliba.it