Congestion Control based on Forward path Status
draft-gwock-rmcat-ccfs-02

Abstract

This document describes CCFS(Congestion Control based on Forward path Status), a rate adaptation scheme for interactive real-time media applications, such as video conferencing. CCFS classifies the forward paths's status as throttled, competing, probing and default which is managed based on estimated parameters - bottleneck bandwidth, queue delay and loss ratio. Considering only forward path status minimizes influence of backward path's network events such as congestion. It is also free from compatibility issues because it defines generalized feedback message.

Table of Contents

1 Introduction

Interactive real-time multi-media applications have a requirement that controls their transmitting rate to adapt to bandwidth changes and maintains low queuing delay over the network [RFC2914]. To solve this challenge, many metrics such as RTT, packet loss and ECN are used to reason the current network condition.

These real-time communication applications can have two streaming paths - forward path to send media and backward path to receive media. Moreover, each path is independent in most of the cases. For example, if congestion occurs in the backward path, their is no need to lower the transmitting rate on the forward path. However, if RTT is used for congestion control, careful approaching is required because RTT could be affected by not only the forward path's latency but the backward path's latency. Although it is used to control the transmitting rate, a metric or behavior such as RTT could be affected by backward path's status and lead to a wrong decision.

CCFS uses metrics reflecting only the forward path's characteristic in its algorithm to remove the influence of backward path's conditions.

CCFS is a sender-based method to be free from compatibility issues. That is, coexistence of multiple CCFS sender versions are available because of algorithm variations or any other issues. To achieve this, passive behavior of CCFS receiver and generalized feedback mechanisms are defined in this memo.

2 Terminology

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

3 Overview

There are two modules to carry out a CCFS - Txer and Rxer. Txer is an abbreviation for transmitter of CCFS and rxer means a receiver of CCFS. Txer has a main active role to control the transmitting bitrate by examining CCFS feedback messages from an associated rxer. Rxer operates passively except when sending periodic CCFS feedback messages. Txer and rxer manage multiple RTP streams if they are able to share network paths. For example, multiple RTP streams using same 4 tuple - source ip, source port, destination ip and destination port - would be associated with one txer and rxer.

To start CCFS, Txer and rxer must complete CCFS negotiation. Negotiation should be accomplished on an external channel before associating RTP session is established.

Rxer sends a periodic CCFS feedback message if CCFS feature is negotiated. Txer estimates various metrics, mainly 3 metrics - bottlenecked bandwidth, queue latency and loss ratio. Then, it makes a decision on the forward path's status, which is one of throttled, probing, competing and default. Txer operates target transmitting rate based on the forward path's status.

Throttled status: detected the network queue is piling up. The current transmitting rate should be lowered to empty the queue.

Competing status: detected cross traffics. The current transmitting rate should be controlled to keep the queue latency within targeting queue latency.

Probing status: The forward path's bottlenecked bandwidth may be larger than the estimated bandwidth. The current transmitting rate should be increased to probe the bottlenecked bandwidth.

Default status: does not belong to above 3 statuses. The current transmitting rate should be controlled to keep the queue latency within targeting queue latency.

While the probing status, the transmitting rate should be increased to probe available bandwidth. However, it can lead to congestion and this can harm the media quality. To minimize the side effects, sending redundant packets like FEC packets is recommended[I-D.ietf-dt-rmcat-adaptive-fec].

4 Detailed Description of CCFS

4.1 CCFS Negotiation

CCFS must be negotiated before run. Defining the way of negotiation is beyond the scope of this document. It may use SDP negotiation[RFC 4566] or an application defined protocol. However, parameters that should be decided from negotiation are described here.

1. txer id (4 bytes): CCFS txer's ID should be decided. This is used as SSRC in RTCP messages used by CCFS. So this value should unique among transmitting RTP stream's SSRC.

2. rxer id (4 bytes): Also, the rxer associated with the txer must have its own ID to use as SSRC in RTCP messages.

3. preferred feedback interval time: Rxer should know initial feedback interval time. This interval may be changed by the txer in the session.

4. preferred monitoring time: A feedback message has information of received packets for this recent monitored time.

5. lower-layer protocol: RTP packets are sent on the UDP stack in most cases. However it may be sent on other transport layers dependding on the application requirement. A different congestion control mechanism for different lower-layer protocol stack is reasonable. To decide which congestion control mechanism should be used, both rxer's transport layer and txer's transport layer is needed. CCFS described in this memo supposes only UDP. However CCFS txer may be modified for other transport layers.

4.2 Rxer

Rxer must know the preferred feedback interval time and its default is 100 msec. This feedback interval time can be changed by RTCP message sent by a txer. Rxer starts a periodic timer with that interval when a session starts. When the timer fires an event rxer determines whether sends a feedback or not. Rxer does not send feedbacks if it has not received any packets and has not sent a feedback before, even when the feedback interval time is passed. So the first feedback message must include reports about one or more received RTP packets. And the rxer should periodically send feedbacks once it has sent the first feedbacks message. Even when there is no received packets for the last feedback interval time, the rxer must send a feedback because it could be an important signal.

One rxer is able to report multiple receiving RTP streams that seem to use the same network path into one feedback message. This may make a larger feedback message. If a feedback message size can be bigger than MTU size, immediately rxer must sends the feedback even before arriving the next feedback interval time. This feedback message's monitored time value should be smaller than preferred feedback interval time. Rxer must restart periodic timer without changing the used interval right after sending the large feedback message.

4.2.1 Feedback message format

CCFS feedback message has a similar design goal as the [I-D.ietf-dt-rmcat-feedback-message]. However, CCFS feedback message needs more specific information for the CCFS algorithm.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|  FMT    |   PT = 205    |          length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SSRC (Rxer Id)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SSRC (Txer Id)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                 CCFS Feedback Message Header                  +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
.              Report-Block of 1st Media Source                 .
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
.              Report-Block of Nth Media Source                 .
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

CCFS Feedback Message Format

The first 4 octets are the RTCP header, with PT=205 and FMT=CCFSFB, and next 4 octets are the SSRC of the packet sender. CCFS replaces SSRC with rxer id which should be obtained from the prior negotiation.

Section 6.1 of [RFC4585] requires the RTCP header to be followed by the SSRC of the media source being reported upon. Txer id is located here instead of a specific RTP SSRC. And SSRC of the media source to be reported is located within its Report-Block.

And next 8 octets are a CCFS Feedback Message Header that is formatted as below:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Report Time                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Feedback Sequence       |      Monitored Time           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

CCFS Feedback Message Header

Report Time(4 octets) : The time instant when this feedback message is generated. The value of the report time is derived from the same wall clock used to generate the NTP timestamp field in RTCP Sender Report (SR) packets. It is formatted as the middle 32 bits of an NTP format timestamp, as described in Section 4 of [RFC3550]. If the rxer does not use NTP, it can be replaced with other measures such as system up time, but the corresponding txer should be informed.

Feedback Sequence(2 octets) : Feedback message's own sequence number. Txer finds out the feedback message was lost or not by watching this feedback sequence's increasement. If a feedback message was lost, Txer must ignore the period that is monitored by the lost feedback message. Also Txer can figure out the forward path's packet loss event if the reported packet sequence number is not continued even though the feedback sequence is increase by one.

Monitored time(2 octets) : Actually observed millisecond duration to generate a feedback message. Normally this is the time by txer preferred and equal or bigger than the current feedback interval time. However if the building feedback information causes bigger message size than MTU size, rxer must immediately send the feedback message with the real monitored time. So that monitored time can be smaller than the current txer preferred monitor time.

Then multiple report blocks are followed. One report block describes received packets from one media source that is identified by SSRC. Report block size is not fixed and format is here:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    SSRC of a Media Source                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Report Packet Count      |    Start Sequence Number      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Pkt-Element_1 | Pkt-Element_2 |             ...               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Pkt-Element_n |              Zero Padding                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

CCFS Report-Block

SSRC of a Media Source(4 octets) : RTP SSRC to report.

Report Packet Count(2 octets) : The reporting rtp packet count. If this value is zero remain fields of this Report-Block should be ignored.

Start Sequence Number(2 octets) : Start RTP sequence number of reported packets. This is the sequence number of the following first Pkt-Elements.

Pkt-Element(1 or 2 octets) : Describes for each packet. The count of Pkt-Element is the total report packet count and these are sorted by RTP sequence number order.

Pkt-Element format is here:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|L|X| Abs-Delta |
+-+-+-+-+-+-+-+-+
```

One octet Pkt-Element. X=0.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|L|X|E|N|      Abs-Delta        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Two octet Pkt-Element. L=0 and X=1.

L(1 bit) : Set for a lost packet. If set, Pkt-Element size is one octet and remain 7 bits must be ignored.

X(1 bit) : Set if Pkt-Element size is two octets.

E(1 bit) : Set if received packet and ECN of IP header of RTP packet is 0x3.

N(1 bit) : Set if packet interval of the received packet is negative. This means the packet was out of ordered.

Abs-Delta(6 or 12 bits) : Packet arrival interval millisecond that is presented in absolute value. The interval is the time between the received time of the lower sequenced RTP packet and the received time of the Pkt-Element's RTP packet.

If the Pkt-Element is the first, the Abs-Delta is an interval from a monitoring start time and it is always positive. And the monitoring start time is the subtracted monitored time from report time.

The Abs-Delta of remain Pkt-Elements is the absolute interval arrival time between two RTP packets. In the most cases, the sequence number of two RTP packets will be successive but it is not always true because of various network conditions. When S sequenced RTP packet(hear in after "S RTP") is lost, Abs-Delta for S+1 RTP means interval time between S-1 RTP and S+1 RTP. Also, S RTP can be arrived before S+1 RTP. In this case, Abs-Delta for S+1 RTP means for interval time between S RTP and S+1 RTP. And N flag for S+1 RTP must be set.

If an absolute value of packet arrival interval is bigger than 63, Pkt-Element size must be 2 octets with X set.

When an absolute value of packet arrival interval is bigger than 4094(2^12 - 2) milliseconds, Abs-Delta must be 4095(2^12 - 1). This means the packet arrival interval is 4095 milliseconds or more.

4.2.2 CCFS feedback message size

Rxer builds a feedback message based on recent received RTP packets. One rxer aggregates multiple RTP streams. And sometimes, by network condition, many packets should be arrived in a short time. These facts make CCFS feedback message packet big. If the CCFS feedback message size is bigger than an accepted packet size by MTU, which will cause exception cases. So CCFS rxer should consider feedback message size. That is, the CCFS feedback message packet size should be smaller than MTU size.

If the feedback message size approaches an available size by MTU, rxer must immediately send the feedback message. The monitored time value within the feedback message will be shorter than a txer preferred monitoring time. After sending the instant feedback message, rxer must re-start monitoring for the next feedback.

CCFS feedback message size can be estimated as:

FBM-Size = 20 + 8R + 1.5T

R means an associated RTP stream count. And T means a total report packet count. 1.5 is the assumed average size of Pkt-Element and this can not be exact. However recommend the constant 1.5 for the simplicity of the implementation.

4.2.3 Handle CCFS control messages

CCFS control message is a type of RTCP message sent by a txer. This RTCP messages should be defined if the txer requires for a specific feature. If the rxer receives understandable control messages, it should respond accordingly. If not, it should ignore and discard them.

4.3 Txer

Txer keeps sent rtp packet array(txed_rtps) about rtp streams. The txed_rtps contains sent local timestamp, packet size, RTP sequence number and ssrc.

Txer estimates variable metrics when the feedback message is sent for each time txer receives a feedback message. This means that all the estimated metrics are the past of backward one way latency ago but remove the effect of the backward path that is the feedback message's network path.

It estimates forward path bandwidth, queue latency and loss ratio using the feedback message and txed_rtps in monitored time.

Feedback messages have its own sequence number. Txer can examine there is forward path packet losses between successive feedback messages. Also when the increase of the feedback message's sequence number is over one, which means backward path packet losses, txer must take consider only reported periods to remove affect of the backward path's network impairments.

And than it decides forward path status and targeting send rate based on the metrics and current status.

The forward path status has four status and described in Section 3. Actually this status affects txer's logic in globally.

4.3.1 Constants

Txer logic is described using pseudo code. For the simplicity, all the constants used are listed up here.

```
FwdBwEstWei = 0.9
I_FwdBwEstWei = (1.0 - FwdBwEstWei)

TargetQDelay = 50msec

WndBrFract = 1sec

TrigProbQDFractMax  = 0.8
TrigProbBrFractMin  = 1.0
TrigProbQMRangeMax  = 10msec
TrigProbLossRtMax   = 0.02
TrigProbECNRtMax    = 0.0

TrigStopProbQDFractMin = 1.3
TrigStopProbBrFractMin = 1.2
```

```
TrigStopProbLossRtMax    = 0.0
TrigStopProbECNRtMax     = 0.0

TrigCompQDelayMin  = 200msec
TrigCompQMRangeMi  = 100msec

TrigStopCompQDelayMax = 100msec
TrigStopCompQMRangeMax = 20msec

TrigThroQDFractMin = 1.5
TrigThroBrFractMax = 0.9

Thro2CompQIncrTime = 1sec

DfltQDFractLow = 0.5
DfltQDFractHi  = 1.1
DfltBrIncrRate  = 1.01
DfltBrDecrRate  = 0.95

CompTargetTxbwRate = 1.3
ThroTargetTxbwRate = 0.5

ProbingTime = 4sec

CompQDFractLow = 0.5
CompQDFractHi  = 1.0
CompBrIncrRate = 1.02
```

4.3.2 Monitoring time on txer

Whenever txer receives a feedback message, txer calculates the monitored time that is matched with the monitored time on rxer.

The latest end sequence in the feedback message is the base point of time to find out monitored time on txer. However total reported packet could be zero that means there was no received rtp packet for the last monitored time for all receiving rtp streams.

In this case, monitoring time on txer must be shifted for monitored time on rxer.

```
--------------------------------------------------------------------
shift_time = fbm.report_time - last_fbm_report_time
end_tstmp = last_end_tstmp + shift_time
bgn_tstmp = end_tstmp - fbm.monitored_time
--------------------------------------------------------------------
```

The fbm stands for feedback message and tstmp is timestamp of txer. This instance has results from parsed a feedback message.

bgn_tstmp is begin timestamp and end_tstamp is end timestamp. They are points of time for a monitored period on txer.

For the more general cases, total reported packet count would be more than zero. Txer determines a monitored period on txer using the latest rtp packet in the reported packets.

```
--------------------------------------------------------------------
pkt_key = (ssrc, latest_end_seq) = find_out(fbm)

latest_sent_tstmp = txed_rtps.get_tstmp(pkt_key)
offset = fbm.report_time - fbm.get_rcvd_time(pkt_key)

end_tstmp = latest_sent_tstmp + offset
bgn_tstmp = end_tstmp - fbm.monitored_time
--------------------------------------------------------------------
```

First of all, find out the sent local time stamp(latest_sent_tstmp) of the latest rtp packet among the reported.

And offset is a time between feedback message report time and the received time of the latest rtp packet.

Each received time for reported rtp packets is calculated as follow:

```
--------------------------------------------------------------------
seq = fbm.start_sequence_number
sum_received_time = fbm.report_time - fbm.monitored_time

for all seq in a SSRC
    received_time[seq] = sum_received_time + fbm.get_delta(seq)
    sum_received_time = received_time[seq]
--------------------------------------------------------------------
```

4.3.3 Forward path bandwidth estimation

The forward path's bandwidth is estimated based on received rate on the rxer.

```
fwd_bw = tot_rxed_size / fbm.monitored_time
```

The tot_rxed_size is sum of sent packet size that is found from txed_rtps with the reported ssrc and seq. If there are the lost packets, they should be excluded. CCFS updates esti_bw with the

fwd_bw using moving average to remove outlier. Unfortunately the moving average calculation causes time penalty. Moreover, if wrong estimated value - too small or too large is used, it would affect esti_bw negatively. So, CCFS checks its validation to minimize the noise.

```
--------------------------------------------------------------------
if( status != Throttled &&
    (status == Competing && target_bw < fwd_bw && lost == 0) &&
    (sent_size > tot_rxed_size)
      || (sent_size == tot_rxed_size && target_bw < fwd_bw) )

    esti_bw = FwdBwEstWei*esti_bw + I_FwdBwEstWei*fwd_bw
--------------------------------------------------------------------
```

First of all, the current status must not be throttled because target bandwidth shrinks during throttled status to empty queue. And if the current status is competing update esti_bw only if it fulfils the certain condition. After status check, sent_size should be larger than tot_rxed_size because it means the sent bitrate is about bottlenecked bandwidth or greater for the period. And if the current target bandwidth is underestimated, it updates esti_bw.

4.3.4 Queue delay estimation and find increase pattern

CCFS uses LEDBAT[RFC6817]'s queue delay estimation to estimate forward path's queue delay. To do that, received timestamp have to be calculated for each packets using ATO field in the feedback message. And the queue delay is the minimum queue delay among the reported packet's estimated queue delays.

The queue delay can not be exact but its relative values and pattern can be used as an important signal. CCFS finds out increasing pattern and its duration as follows.

```
--------------------------------------------------------------------
if( last_qdelay < qdelay )
    incr_count++

    if(incr_count == 1)
        incr_start_tstmp = end_tstmp

    incr_duration = end_tstmp - incr_start_tstmp;

    if(incr_count >= 3 && duration >= IncrMinDuration)
        is_increasing = true
else
    incr_count = 0
```

```
        incr_start_tstmp = 0
        is_increasing = false

    last_qdelay = qdelay
    -------------------------------------------------------------------
```

Above logic can be replaced by others if it shows good performance.

4.3.5 Handle by status

Update transmitting rate (target_txbw) based on the calculated parameters.

```
-------------------------------------------------------------------
qd_fract = qdelay / target_qdelay
br_fract = recved_size_wnd / sent_size_wnd
-------------------------------------------------------------------
```

Above two fractions are used directly to check status and control send rate. The recved_size_wnd means that total received packet size for the last window time(WndBrFract) and the sent_size_wnd is the total sent packet size for the same time.

4.3.5.1 Control event

Before controlling transmitting rate, a certain condition makes status change and CCFS defines this conditions as control event. The control event list and required condition are presented here.

Control Event	Conditions
CTRL_NOTHING	* default value
CTRL_START_COMPETE	1. qdelay > TrigCompQDelayMin && qmrange > TrigCompQMRangeMin 2. Throttled status && incr_duration > Thro2CompQIncrTime
CTRL_START_PROBING	status is not Probing && is_increasing == false && qd_fract < TrigProbQDFractMax && br_fract >= TrigProbBrFractMin && qmrange < TrigProbQMRangeMax && ecn_rate < TrigProbECNRtMax && loss_rate < TrigProbLossRtMax
CTRL_DETECT_THROTTLE	is_increasing && qd_fract > QDFractMinTrigThro && br_fract < BrFractMaxTrigThro
CTRL_STOP_COMPETE	Competing status && comp_duration > CompMaintainTime && qdelay < QDelayMaxTrigCompStop && qmrange < QMRangeTrigCompStop
CTRL_STOP_PROBING	1. Probing status && is_increasing 2. Probing status && qd_fract > TrigStopProbQDFractMin 3. Probing status && br_fract > TrigStopProbBrFractMin 4. Probing status && ecn_rate >= TrigStopProbECNRtMax 5. Probing status && loss_rate >= TrigStopProbLossRtMax
CTRL_RESOLV_THROTTLE	Throttled status && qdFract < 1.0

4.3.5.2 **Handle control event by status**

CTRL_START_COMPETE and CTRL_DETECT_THROTTLE are important signals to react promptly irrelevant the forward status. So, extracted handlers are described as follows.

```
--------------------------------------------------------------------
do_start_compete():
    target_qdelay = TargetQDelay + TargetXQDelay
    target_txbw = esti_bw * CompTargetTxbwRate
    status = Competing
return

do_detect_throttle():
    thro_backup_target_txbw = esti_bw
    target_txbw = esti_bw * ThroTargetTxbwRate
    status = Throttled
return
--------------------------------------------------------------------
```

Comprehensive handling for each status may seem to be complicated. So, this document supplements the pseudo code with simple available status change diagram.

```
         +-------------------+                      +---------+
         |      Default      |<====================>| Probing |
         +-------------------+                      +---------+
           | ^        | ^                                |
           | |        | |                                |
           | |        | |                                |
           | |        | |                                |
           | |        V |                                |
           | |   +-----------+                           |
           | |   | Competing |<--------------------------+
           | |   +-----------+                           |
           | |        | ^                                |
           | |        | |                                |
           | |        | |                                |
           | |        | |                                |
           V |        V |                                |
         +-------------------+                           |
         |     Throttled     |<--------------------------+
         +-------------------+
```

```
+----------------+
| Default status |
+----------------+
Event: CTRL_NOTHING
    if(qd_fract < DfltQDFractLow && target_txbw < esti_bw)
        target_txbw *= DfltBrIncrRate
    else if(qd_fract > DfltQDFractHi)
        target_txbw *= DfltBrDecrRate

Event: CTRL_START_PROBING
    prob_backup_target_txrt = esti_bw
    target_txbw = esti_bw + prob_bw
    prob_start_tstmp = curr_tstmp
    status = Probing

Event: CTRL_START_COMPETE
    do_start_compete()

Event: CTRL_DETECT_THROTTLE
    do_detect_throttle()

+------------------+
| Competing status |
+------------------+
Event: CTRL_NOTHING
    if( qd_fract < CompQDFractLow || qd_fract < CompQDFractHi )
        target_txrt *= CompBrIncrRate

Event: CTRL_SOTP_COMPETE
     target_qdelay = TargetQDelay
     status = Default

Event: CTRL_DETECT_THROTTLE
     do_detect_throttle()
```

```
   +----------------+
   | Probing status |
   +----------------+
   Event: CTRL_NOTHING
       if(curr_tstmp - prob_start_tstmp > ProbingTime)
           status = Default
           target_txnw = prob_backup_target_txbw + prob_bw

   Event: CTRL_STOP_PROBING
       target_txbw = esti_bw
       status = Default

   Event: CTRL_START_COMPETE
        do_start_compete()

   Event: CTRL_DETECT_THROTTLE
        do_detect_throttle()

   +------------------+
   | Throttled status |
   +------------------+
   Event: CTRL_RESOLV_THROTTLE
        target_txbw = thro_backup_target_txbw
        status = Default

   Event: CTRL_START_COMPETE
        do_start_compete()

   Event: CTRL_DETECT_THROTTLE
        thro_backup_target_txbw = esti_bw
        target_txbw = esti_bw * ThroTargetTxbwRate
```

4.4 CCFS Control messages

If txer wants to implement a specific feature that needs rxer's help, it can send CCFS control messages. CCFS control message is a RTCP message with FMT=CCFSCTRL value.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|  FMT    |   PT = 205    |          length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SSRC (Txer Id)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SSRC (Rxer Id)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X|  CMT    |   Length          |          . . .                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
.                      Control Message Value                    .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SSRC for media source is replaced with Rxer Id.

Control message block is followed and it can be multiple. The X bit indicates there is a control message block following the current block.

CMT(5 bits) is the control message type to inform rxer which control message value type should be used. If rxer does not understand the CMT, it can discard the message.

Length(10 bits) is the octet size of the control message value.

Control Message Value is different depending on the CMT value.

4.4.1 Update preferred feedback interval

Txer can change the preferred feedback interval if need. This message doesn't need to be guaranteed so rxer won't send response message.

CMT: 1
Length: 2
Control Message Value: Interval time(msec)

4.4.2 Update preferred monitoring time

Txer can change the monitoring time if need. This message doesn't need to be guaranteed so rxer won't send response message but applied feedback message have the updated monitored field value.

CMT: 2
Length: 2
Control Message Value: Monitoring time(msec)

5 Implement

<TBD>

6 Security Considerations

<Security considerations text>

7 IANA Considerations

<IANA considerations text>

8 References

8.1 Normative References

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

[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>.

[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>.

[KEYWORDS] 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>.

[RFC1776] Crocker, S., "The Address is the Message", RFC 1776, DOI 10.17487/RFC1776, April 1 1995, <http://www.rfc-editor.org/info/rfc1776>.

[TRUTHS] Callon, R., "The Twelve Networking Truths", RFC 1925, DOI 10.17487/RFC1925, April 1 1996, <http://www.rfc-editor.org/info/rfc1925>.

8.2 Informative References

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <http://www.rfc-editor.org/info/rfc2914>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC5513] Farrel, A., "IANA Considerations for Three Letter Acronyms", RFC 5513, DOI 10.17487/RFC5513, April 1 2009, <http://www.rfc-editor.org/info/rfc5513>.

[I-D.ietf-dt-rmcat-feedback-message] "RTP Control Protocol (RTCP) Feedback for Congestion Control", <https://tools.ietf.org/html/draft-ietf-avtcore-cc-feedback-message-02>

[I-D.ietf-dt-rmcat-adaptive-fec] "Congestion Control Using FEC for Conversational Media", <https://datatracker.ietf.org/doc/draft-singh-rmcat-adaptive-fec/>

Authors' Addresses

Jungnam Gwock
Line Plus
South Korea

EMail: jungnam.gwock@linecorp.com

Sanghyun Lee
Line Plus
South Korea

EMail: sanghyun.lee@linecorp.com