Congestion Control based on Forward path Status

draft-gwock-rmcat-ccfs-00

Abstract

This document describes CCFS(Congestion Control based on Forward path Status), rate adaptation scheme for interactive real-time media applications, such as video conferencing. CCFS manages forward path’s status based on the estimated three major parameters - serving bitrate, queue delay and queue delay by cross traffics. Considering only forward path status minimizes the effects of backward path’s network event such as congestion. It also is free from compatibility issues because it defines generalized feedback message and minimizes receiver’s role.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2018 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1 Introduction ......................................................... 3
2 Terminology ....................................................... 3
3 Overview ............................................................ 3
4 Detailed Description of CCFS ....................................... 5
   4.1 CCFS Negotiation ............................................... 5
4.2 Rxer ............................................................... 5
   4.2.1 Feedback message format .................................... 5
   4.2.2 Handle CCFS control messages ............................... 7
4.3 Txer ............................................................... 7
   4.3.1 Constants ....................................................... 7
   4.3.2 Monitored time on txer ........................................ 9
   4.3.3 Forward path bandwidth estimation .......................... 9
   4.3.4 Queue delay estimation and find increase pattern ........... 10
   4.3.5 XQueue delay : Queue delay by external traffics ........... 10
   4.3.6 Handle by status ................................................ 11
       4.3.6.1 Control event ............................................. 11
       4.3.6.2 Handle control event by status ............................ 12
4.4 CCFS Control messages ........................................... 15
   4.4.1 Update feedback interval ..................................... 15
5 Implement ............................................................ 15
6 Security Considerations ............................................ 16
7 IANA Considerations ................................................ 16
8 References .......................................................... 16
   8.1 Normative References ............................................ 16
   8.2 Informative References ......................................... 16
Authors’ Addresses ................................................... 17
1 Introduction

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

These real-time communication application can have two streaming path - forward path to send media and backward path to receive media. Moreover, each path is independent. For example, if congestion occurs in the backward path, there is no need to lower transmitting rate on the forward path. However if the 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 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 character in its algorithm to remove the effect of backward path’s conditions.

Moreover, CCFS is sender-based method to be free from compatibility issues. That is, coexistence of multiple CCFS sender versions are available because of performance 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. The txer is an abbreviation for transmitter of CCFS and rxer means receiver of CCFS. The txer have a main active role to control transmitting bitrate by examining CCFS feedback messages from rxer. The rxer operates passively except when sending periodic CCFS feedback messages. The txer and rxer manage multiple RTP streams if they are able to share network paths. For example, if any 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, the txer and rxer must complete CCFS negotiation. The
negotiation should be accomplished on an external channel before the associating RTP session is established.

The rxer sends a periodic CCFS feedback message if CCFS feature is negotiated. The txer estimates various metrics, mainly 3 metrics – bottlenecked bandwidth, queue latency and queue latency from external traffics. Than, it makes a decision about the forward path’s status, which is one of Default, Probing, Competing and Throttled. Txer operates based on the resulting forward path’s status.

Throttled status: detected the network queue is piling up. The current transmitting rate should 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 broader 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. feedback interval time: Rxer should know initial feedback interval time. This interval may be changed by the txer in the session.

4. lower-layer protocol: RTP packets are sent on the UDP stack in most cases. However it may be sent on other transport layers depending 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 monitors received rtp packets and send feedback on every feedback interval. That is, a rxer uses only the periodic feedback mechanism. The feedback interval time can be changed by RTCP message sent by a txer.

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. However, the rxer should periodically send feedbacks once it has started sending feedbacks even when there is no received packets for the last feedback interval because it could be an important signal.

4.2.1 Feedback message format

CCFS feedback message has a similar design goal as the [I-D.ietf-dtmcat-feedback-message]. However, CCFS feedback message needs more specific information for the CCFS algorithm.
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 because this feedback message cannot designate one SSRC.

And next 8 octets are a kind of the header of the CCFS feedback message. The report time is the time instant when this feedback message is generated. The value of the report time is derived from the same wallclock 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 is to feedback its own sequence and inform txer where the rtp packet loss has occurred when the rtp packet between feedback messages has been lost. And monitored time follows. The feedback interval can change with the txer’s request.

Then, report blocks are followed. A report block starts with the SSRC of that media source. The reporting rtp packet count and end sequence number of the reporting rtp packets are followed. Packet metric blocks of count indicated by report packet count are followed. Each packet metric block is 2 octets. The L, ECN and ATO fields are the same with the [I-D.ietf-dt-rmcat-feedback-message]

4.2.2 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 seq number and ssrc.

Txer estimates variable metrics when the feedback message is sent for each time rxer 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 queue latency from external traffics with the feedback message and txed_rtps in monitored time.

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)

MaXQWei = 0.9
I_MaXQWei = (1.0 - MaXQWei)

TargetQDelay = 50msec
TargetXQDelay = 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
CompXQDFractLow = 0.8
CompBrIncrRate = 1.02

4.3.2 Monitored time on txer

When every time rxer 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.

(latest_end_seq, ssrc) = find_out(fbm)

sent_tstmp_end_seq = get_tstmp(txed_rtps, latest_end_seq, ssrc)

end_tstmp = sent_tstmp_end_seq + fbm.ato(latest_end_seq, ssrc)

bgn_tstmp = period_end_tmstmp - fbm.monitored_time

The fbm stands for feedback message and tstmp is time stamp of txer.

First of all, find the sent local time stamp(sent_tstmp_end_seq) of the latest rtp packet among the reported. And the monitored end time on txer sets as the ato time and sent_tstmp_end_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 fulfills 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(inchr_count == 1)
    incr_start_tstmp = end_tstmp

  incr_duration = end_tstmp - incr_start_tstmp;

  if(inchr_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 XQueue delay : Queue delay by external traffics

The queue delay by external traffics is calculated by subtracting expected queue delay of sent packets from queue delay.

\[
\text{exp_qdelay} = \frac{\text{sent_size}}{\text{esti_bw}} \\
\text{xqdelay} = \text{qdelay} - \text{exp_qdelay}
\]
The xqdelay could be negative because of an instant high sent_size or under estimated esti_bw. Moreover, the xqdelay can have outlier too. So, xqdelay applies a simple filter as follows.

--------------------------------------------------------------------
if (xqdelay > 0)
  if(maxq == 0.0) maxq = xqdelay
  else            maxq = MaxQWei*maxq + I_MaxQWei*xqdelay
else
  maxq = 0.0
--------------------------------------------------------------------

4.3.6 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
xq_fract = 0.0

if (maxq)
  xq_fract = xqdelay / TargetXQDelay

--------------------------------------------------------------------
Above three 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.

4.3.6.1 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.

<table>
<thead>
<tr>
<th>Control Event</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL_NOTHING</td>
<td>* default value</td>
</tr>
</tbody>
</table>
| CTRL_START_COMPETE    | 1. qdelay > TrigCompQDelayMin
                           && qmrange > TrigCompQMRangeMin
                           && xq_fract > 0.0
                           2. Throttled status
                           && incr_duration > Thro2CompQIncrTime |
4.3.6.2 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.

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

do_detect_throttle():
```
thро_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.

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 && xq_fract > CompXQDFractLow) )
target_txt *= 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.

|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 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 is the octet size of the control message value.

Control Message Value is different depending on the CMT value.

4.4.1 Update feedback interval

Txer can change the feedback interval 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: 1
Length: 2
Control Message Value: Interval time(msec)

5 Implement
6 Security Considerations

<Security considerations text>

7 IANA Considerations

<IANA considerations text>

8 References

8.1 Normative References


8.2 Informative References

J. Gwock Expires March 7, 2019
Internet Draft                    CCFS                       August 2018

[RFC2914]  Floyd, S., "Congestion Control Principles", BCP 41, 
            RFC 2914, DOI 10.17487/RFC2914, September 2000, 

            Description Protocol", RFC 4566, DOI 10.17487/RFC4566, 

[RFC5513]  Farrel, A., "IANA Considerations for Three Letter 
            Acronyms", RFC 5513, DOI 10.17487/RFC5513, April 1 2009, 

[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://tools.ietf.org/html/draft-singh-rmcat-adaptive- 
fec-03>

Authors’ Addresses

Jungnam Gwock
Line Plus
South Korea

EMail: jungnam.gwock@linecorp.com