**3GPP SA4 #128 S4-241058**

**Jeju, Korea, 20 May-May 24 2024**

|  |
| --- |
| *CR-Form-v12.0* |
| **PSEUDO CHANGE REQUEST** |
|  |
|  | **26.822** | **CR** | pseudo | **rev** | **-** | **Current version:** | **0.0.1** |  |
|  |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network |  | Core Network |  |

|  |
| --- |
|  |
| ***Title:***  | **[FS\_5G\_RTP\_Ph2] Real-time Communication Congestion Control Algorithms and AL-FEC**  |
|  |  |
| ***Source to WG:*** | Qualcomm Incorporated |
| ***Source to TSG:*** |  |
|  |  |
| ***Work item code:*** | FS\_5G\_RTP\_Ph2 |  | ***Date:*** | 05/20/2024 |
|  |  |  |  |  |
| ***Category:*** | **B** |  | ***Release:*** | Rel-19  |
|  | *Use one of the following categories:****F*** *(correction)****A*** *(mirror corresponding to a change in an earlier release)****B*** *(addition of feature),* ***C*** *(functional modification of feature)****D*** *(editorial modification)*Detailed explanations of the above categories canbe found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | *Use one of the following releases:Rel-10 (Release 10)Rel-11 (Release 11)Rel-12 (Release 12)**Rel-13 (Release 13)Rel-14 (Release 14)Rel-15 (Release 15)Rel-16 (Release 16)* *Rel-17 (Release 17)* *Rel-18 (Release 18)* |
|  |  |
| ***Reason for change:*** | One aspect of Key issue #4: AL-FEC awareness for PDU Set handling:“Furthermore, there are intricate interactions to consider between the application and the network.” To address the above aspect, we need to understand how existing real-time communication congestion control algorithms interact with the network.  |
|  |  |
| ***Summary of change:*** | Added descriptions of congestion control algorithms deoployed in commercial systems (WebRTC) and documented in IETF.Added obervations on how existing real-time communication congestion control algorithms interact with the network. |
|  |  |
| ***Consequences if not approved:*** | Lack of understanding of the interactions between the application and the network.  |
|  |  |
| ***Clauses affected:*** |  |
|  |  |
|  | **Y** | **N** |  |  |
| ***Other specs*** |  |  |  Other core specifications  | TS/TR ... CR ...  |
| ***affected:*** |  |  |  Test specifications | TS/TR ... CR ...  |
| ***(show related CRs)*** |  |  |  O&M Specifications | TS/TR ... CR ...  |
|  |  |
| ***Other comments:*** |  |
|  |  |
| ***This CR's revision history:*** |  |

.

#  Proposed changes

\* \* \* \* 1st change \* \* \* \*

Add the following to the References clause:

## [GCC-IETF] A Google Congestion Control Algorithm for Real-Time Communication, draft-ietf-rmcat-gcc-02, 2016.

[WebRTC-code] WebRTC source code: [https://source.chromium.org/chromium/chromium/src/+/main:third\_party/webrtc](https://source.chromium.org/chromium/chromium/src/%2B/main%3Athird_party/webrtc), retrieved May 1, 2024.

[NADA] RFC 8698, Network-Assisted Dynamic Adaptation: A Unified Congestion Control Scheme for Real-Time Media, 2020.

[SCReAMv2] Self-Clocked Rate Adaptation for Multimedia, draft-johansson-ccwg-rfc8298bis-screamv2-00, 2024.

\* \* \* \* End of 1st change \* \* \* \*

\* \* \* \* 2nd change \* \* \* \*

## 5.4.x Real-time Communication Congestion Control Algorithms and AL-FEC

We introduce the congestion control algorithms for real-time communication, including those currently implemented in WebRTC [WebRTC-code] and two algorithms described in two IETF RFCs [NADA] [SCReAMv2].

Observations are made on these algorithms and AL-FEC.

### 5.4.x.1 Google Congestion Control (GCC)

Google Congestion Control (GCC) is one of the two congestion control algorithms supported in the libwebrtc implementation of WebRTC. The algorithm determines a sending bit rate, which limits the total bit rate for RTP packets, RTCP packets and RTP retransmissions (if any) and drives the bit rate adaptation of the media codecs. It uses packet losses and delays as signals of network congestion in adjusting the sending rate. The packet loss feedback is based on RTCP reports. The delays are one-way delays measured by the “absolute send time” RTP header extension as described in TS 26.522, and the change in the one-way delay is computed to detect network congestion.

GCC has not been standardized. Although the informational IETF document [GCC-IETF] describes GCC, the description is outdated. In what follows, we present GCC based on the current libwebrtc implementation [WebRTC-code].

GCC sets the sending rate to the lesser of a loss-based bandwidth estimation and a delay-based bandwidth estimation. That is, the sending rate at time is set to:

where is the loss-based bandwidth estimation made at time when the lastest RTCP loss report (the *k*th) is received, and is delay-based bandwidth estimation made at time when the lastest one-way delay measurement is taken or reported.

**Delay-based bandwidth estimation:**

This is described in aimd\_rate\_control.cc, based on the detection of overuse. The one-way delay

where is a constant equal to 1200 bytes/second/RTT, is a backoff factor. The state ‘decrease’, ‘hold’ and ‘increase’ are based on the slope of the change in the one-way delay as a function of time. The change in the one-way delay from packet group to packet group is defined as

where is the arrival time of packet group , and is the departuture time of packet group , is the arrival time of packet group , and is the departuture time of packet group . The slope (termed trendline in the WebRTC source code) is estimated and compared with thresholds to determine the state.

**Loss-based bandwidth estimation:**

There are three versions of loss-based bandwidth estimation algorithms.

**Version 0 (static thresholds):**

The loss-based bandwidth estimation is calculated as:

where is the time when the previous RTCP loss report is received, is the loss (fration of loss) received in the most recent RTCP loss report, and is the loss-based bandwidth estimation at the time when the previous RTCP loss report is received.

**Version 1 (dynamic thresholds):**

This version is similar to version 0, except that it uses (1) dynamic thresholds are used, and (2) the factor for increasing the estimation is adaptive to RTT, instead of being a constant 1.08 in Version 1. The loss-based bandwidth estimation is:

where and are dynamic thresholds, ,

 is the probability for decreasing the bandwidth estimation, equal to the minimum of the average loss probability and the most recent loss probability,

 is the probability for increasing the bandwidth estimation, equal to the average loss probability if and equal to ,

 where is time elaspsed, is a time window,

 is the bit rate of the acknowledged transmissions,

where is the RTT, is a pre-configured minimum RTT and is a pre-configured maximum RTT. With this dynamic threshold , the increase in loss-based bandwidth estimation will be slower when the RTT gets higher.

**Version 2 (maximum likelihood):**

To calculate the loss-limited bandwidth, in a nutshell, the sender chooses a tuple of an inherent loss probability (loss probability induced by the channel error rather than network congestion) and a loss limited bandwidth that maximizes the following objective function which is based on the logarithmic maximum likelihood of observing the number of packet losses:

where:

is the number of observations,

=0.9 is a weight,

 is the number of bytes lost,

 is the number of bytes received,

,

 where is the sending rate at the time of the *i*th observation,

 is a high bandwidth bais (that depends on ),

 is the index of the most recent observation and is the index of the oldest observation.

### 5.4.x.2 PCC

Performance-oriented Congestion Control (PCC) is the other congestion control algorithm supported in the current WebRTC implementation [WebRTC-code]. The sender adjusts the sending rate and observe the performance metrics including the delay and packet loss, and pick the sending rate that maximizes a utility function:

where is the sending rate, is the gradient of RTT , is the loss rate, and are coefficients.

### 5.4.x.3 NADA

Network-Assisted Dynamic Adaptation (NADA) is specified in RFC 8698 [NADA]. This algorithm considers delay, packet loss, and ECN marking as signals of network congestion. Furthermore, it converts packet loss and ECN marking to some equivalent penalty in terms of delay and forms an aggregate congestion signal. Specifically, the receiver calculates

where is the equivalent delay after non-linear warping,

is the estimated packet ECN marking ratio, is the reference packet ECN marking ratio,

 is the eference delay penalty for ECN marking when packet marking is at ,

 is the estimated packet loss ratio, is the reference packet loss ratio,

 is the eference delay penalty for packet loss when packet loss is at .

The receiver decides whether the sender is to be in the accelerated ramp-up mode (rate update mode ) or in the gradual update mode (). The sender should be in the accelerated ramp-up mode if there are no recent packet losses in an observation window (of 500ms) and there is no build up of queueing delay. Otherwise, the sender should be in the gradual update mode.

The receiver also calculates the receiving bitrate

The receiver sends , , and to the sender.

The sender performs multiplicative increase in the accelerated ramp-up mode using the RTT and , and performs additive decrease in the gradual update mode using .

### 5.4.x.3 SCReAMv2

Self-Clocked Rate Adaptation for Multimedia 2 (SCReAMv2) [SCReAMv2] uses packet losses, delay, and ECN marking as signals of network congestion.

A congestion window size limits the sending rate and is adjusted based on:

* **Packet losses:** A packet loss causes the congestion window size to decrease, but not as much as a packet loss does in the case of TCP Reno.
* **Queueing delay:** The congestion window size decreases linearly when the average queueing delay exceeds a threshold.
* **ECN marking:** When a classic ECN marking is received, the congestion window size decreases by a factor. When L4S ECN marking is received, the congestion window size decreases in proportion to the fraction of packets that are L4S ECN marked.

### 5.4.x.4 Summary of congestion control algorithms

The interactions between the application and the network are through signaling manifested by packet losses, queueing delay and ECN marking, as summarized in the table below:

Table 5.4-1

|  |  |  |  |
| --- | --- | --- | --- |
| Congestion Control Algorithm | React to packet losses? | React to ECN marking? | React to queueing delay? |
| GCC (with loss based bandwidth estimation v0) | No, if the packet loss rate is between 2% and 10%;yes otherwise.  | No | Yes |
| GCC (with loss based bandwidth estimation v1) | Yes | No | Yes |
| GCC (with loss based bandwidth estimation v2) | Yes | No | Yes |
| PCC | Yes | No | Yes |
| NADA | Yes | Yes | Yes |
| SCReAMv2 | Yes | Yes | Yes |

**Observation 1:** All congestion control algorithms for real-time communication in Table 5.4-1 use queuing delay (among other metrics) as a signal of network congestion.

**Observation 2:** Although all congestion control algorithms for real-time communication in Table 5.4-1 use packet losses as a signal of network congestion, one algorithm is not sensitive to packet losses when the packet loss rate is within the range [2%, 10%].

**Observation 3**: Two of the six congestion control algorithms for real-time communication in Table 5.4-1 support ECN marking.

Although WebRTC currently implements only GCC (with three versions) and PCC, it does not prevent one from adding other congestion control algorithms such as NADA and SCReAMv2. Since these algorithms support ECN marking, we have the following observation.

**Observation 4:** It is possible to add ECN marking support to WebRTC for congestion control. Updates of RFCs is required to ensure interoperability.

### 5.4.x.5 Packet loss rate calculation for AL-FEC

When AL-FEC is used, the source packets and the repair packets may be sent in a single RTP stream (identified by an SSRC). This is the case for ULPFEC in the WebRTC implementation [WebRTC-code].

The source packets and the repair packets may be sent in different RTP streams (identified by different SSRC’s) within the same RTP session. This is the case for FlexFEC in the WebRTC implementation [WebRTC-code]. The packet loss rate is calculated individually. The packet loss rates are then combined to form a single packet loss rate as an input to the congestion control algorithms.

**Observation 5:** For AL-FEC, the packet loss rate on the source packets and that on the repair packets can be calculated separately. RFC 5109 (Clause 12) give congestion considerations. However, there are no considerations to handle repair losses differently than source packet losses.

Although RFC 5109 [13] recommended that the source packets and the repair packets may be sent in different sessions (identified by different IP 5-tuples), we are unaware of any commercial implementation of such scheme.

Note that RFC 8085 (UDP usage Guidelines) recommend that an “application SHOULD perform congestion control over all UDP traffic it sends to a destination, independently from how it generates this traffic”.

\* \* \* \* End of 2nd change \* \* \* \*