Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 21, 2017

I. Johansson
Ericsson AB
July 20, 2016

Congestion control for 4G and 5G access
draft-johansson-cc-for-4g-5g-02

Abstract

This memo outlines the challenge that 4G and 5G access brings for transport protocol congestion control and also outlines a few simple examples that can improve transport protocol congestion control performance in 4G and 5G access.

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].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Wireless access is becoming more and more widely used, 4G (LTE) access is one wireless access technology that has built in support for seamless mobility that gives the end user a feeling of being always connected. Transport endpoints may even be unaware of the existence of the 4G access. Everyday use for 4G access includes web, chat, streaming video and lately also WebRTC. These use cases pose different requirements on the underlying access. Evolving existing radio-access technologies like LTE, and new 5G technologies will all be part of a future flexible and dynamic 5G system. 5G has potential to offer lower latency and higher peak throughput. The goal of this document is to provide sufficient input to guide the development of congestion control that is better suited for 4G and 5G access, without an explicit need to know about the presence of 4G or 5G access along the transmission path.

2. The 4G protocol stack impact on transport protocols

This section will go into the different layers in the 4G protocol stack. It will not delve in the very details, recommended reading for more details is found in [LTE]. Rather this section will illustrate what effect each layer can have on the transport protocol efficiency. The description is focused mainly on default radio access bearers, which are commonly used for OTT (Over The Top) services, these bearers typically use Acknowledged Mode (AM), which means that packet loss only occurs as the result of packet drops in AQM (Active Queue manager). Specialized services like VoLTE (Voice over LTE) use different bearer configurations, this is however outside the scope of this document. The concept of bearer is mentioned throughout the document, a bearer is to be seen as a data channel for a given terminal or UE (User Equipment), there could be many bearers with different priorities for a given terminal.

```
------------
|          |
|    IP    |
|          |
------------
|   PDCP   |
------------
|   RLC    |
------------
|   MAC    |
------------
|   PHY    |
------------
```

LTE protocol stack

2.1. PDCP layer

The PDCP (Packet Data Convergence Protocol Layer), ensures that intra-RAT (Radio Access Type) handover, i.e. LTE to LTE is reasonably seamless. The PDCP layer makes sure that packets pending transmission in one cell to terminal connection, are transmitted in the new cell to terminal connection. This way all packets will be ensured to be delivered to the endpoint. PDCP also ensures that all packets are delivered in order up to higher layers.

Packet retransmission typically means that the amount of data to transmit increases immediately after the handover, first the retransmission data needs to be transmitted, then the incoming data needs to be transmitted. Depending on the available link throughput after the handover, an increased RTT may be experienced at a handover

event. In addition, a small temporal delay increase can occur as packet transmission is inhibited at the handover event. Unnecessary, retransmission at handover can be minimized by means of PDCP status reports, but this is not always implemented. Because of the above it is a good practice to keep the amount of data in flight as small as possible, without sacrificing throughput. Excessive amounts of data in flight means potentially more data to retransmit at handover and thus an even more increased RTT.

Packets are typically not lost at handover in a 4G/LTE system. Reliable delivery at handover may however be turned off or is simply not implemented, this means that packets may be lost at handover. The amount of lost packets is then proportional to the number of packets in flight, it is therefore instrumental that the amount of packets in flight is as small as possible, with the objective to reach full link utilization, nothing more. Bufferbloat [REF] can for instance lead to that 1000s of packets are lost at handover, which is of course undesired as it can affect media quality quite considerably.

2.2. RLC layer

The role of the RLC (Radio Link Control) layer is to ensure that packets are delivered in order up to the higher layers, in addition the RLC layer corrects errors that can occur on the MAC layer. The in order delivery constraint means that additional HoL (Head of Line) blocking delay can occur due to errors on the MAC layer.

The throughput on lower layers can vary quite considerably, this manifests itself as a varying size of the available transport. The transport block size depends on how much of the available resources is allocated to a given bearer at a given time instant and also on the channel quality. Because of this the size of the transport blocks can vary from tens of bytes, up to more than 10000 bytes. The rate of change in transport block size also varies with terminal mobility as higher terminal mobility means faster changing channel fading and thus a faster changing channel quality.

For optimal spectrum efficiency, it is important that a sufficient amount of data is available to fill the transport blocks, this data needs to be available already when a bearer is scheduled, in practice within a fraction of a millisecond. To satisfy this requirement, packets need to be queued up and ready for immediate transmission, either on the RLC or the PDCP layer. The transport protocol server (TCP, QUIC) is typically too far away to satisfy this need.

The need to instantly provide sufficient data for optimal spectrum efficiency, given the variability in transport block sizes and

scheduling opportunities for a given bearer, quite naturally leads to a variation in queuing delay and this variation can be larger than what is to be expected from e.g. fixed line access.

The requirement above to have data available can be seen something that contradicts the strive for low latency, and there is indeed a balance to be struck here. What to aim for, maximum throughput or low latency, is something that depends on the requirements from the application. A desire for very low latency comes generally at the cost of reduced peak throughput, this applies to default radio access bearers. QoS classed bearers can have different characteristics and may well be able to deliver both very low latency and high throughput.

2.3. MAC layer

The MAC (Media Access Control) layer handles transmission of transport blocks, the outcome of a transmission attempt can be either success or failure. The signaling of the success is handled with a single ACK/NAK bit. Upon indicated failure, the transport block is retransmitted (with a different channel coding), soft decoding is utilized and the softbits of the first transmission and the retransmission are combined, hopefully with a successful result, if not the case a 3rd retransmission can occur and so on. This is referred to as HARQ (Hybrid Automatic Repeat Request). The maximum number of retransmissions is configurable, if the maximum number of retransmission is reached without successful transmission then the RLC layer will have to handle the retransmission instead.

Errors may also happen in the transmission of the ACK/NAK bit. The event that ACK is decoded as NAK will only lead to that an extra superfluous retransmission occur. The event that a NAK is decoded as an ACK will be handled by the RLC layer as the result of a detected RLC checksum error.

MAC layer retransmissions naturally lead to out of order delivery up to higher layers as some transport blocks are transmitted error free while others need retransmission. The role of the RLC layer is to ensure in order delivery, the effect of this is that HARQ retransmissions and HARQ failures lead to additional delays.

Scheduling of many bearers has the effect that available resources have to be shared between two or more bearers. When new bearers with data to transmit are added in a cell (either handover, or new traffic), it means that the amount of resources need to be shared with an additional party. This can give a large drop in available throughput for already existing users, with the effect of an

increased queuing delay that decreased only when the transmission rate over the bearer is reduced.

The downlink and uplink scheduling differs in some details which are described in the following sub-sections.

2.3.1. Downlink scheduling

Downlink scheduling, or scheduling of packets to terminals, is controlled by the base station. For each TTI (1ms interval) a decision is made on which bearer is to become scheduled, i.e. packets are forwarded from the queue to the terminal in question. The scheduling decision is based on channel quality, and possibly historic bitrate for the given bearers, or it may be just a simple round robin scheduler. The very details of the scheduling algorithms are vendor specific.
DRX (Discontinuous Reception) is a feature implemented to save battery power, in which the terminal sleeps and only checks for the presence on downlink data only at regular intervals.
Given the facts above, downlink data cannot always be transmitted immediately, this has the effect that additional jitter may be added (in the order of 10-20ms). Congestion control algorithms that are tuned with a high sensitivity to delay can by mistake treat this jitter as congestion.

2.3.2. Uplink scheduling

As is the case with downlink scheduling, uplink scheduling is controlled by the base station. A terminal that has data to transmit will first transmit a scheduling request to the base station. The scheduling request does not indicate how many bytes that are in the uplink queue. The base station transmits a scheduling grant back, with a delay that depends on the overall load level. The scheduling grant indicates how many bytes that can be transmitted by the terminal. After this the terminal can transmit the allowed number of bytes, if there is still data in the queue, then a BSR (Buffer Status Report) is attached to the uplink transmission which will in turn trigger an additional scheduling grant from the base station to the terminal and so on until all the data in the queue is transmitted.
The uplink scheduling regime outlined above can break up packet trains, for instance a set of 10 ACKs in the uplink may be broken up to an initial transmission of 2 ACKs followed by the transmission of the remaining 8 ACKs, the HARQ RTT is typically 8ms, which means that the remaining 8 ACKs are delivered upstream 8ms later. This can cause problems for congestion control algorithms that depend on e.g. packet train based estimation of throughput. Also, algorithms that depend on precise RTT estimates may by mistake treat the occurrence above as emerging congestion in the downlink.

This behavior can also trigger coalescing issues similar to those experienced when ACK compression occur.
Worth notice is also that the above effects can occur at low as well as high network load levels.

ACK traffic in uplink can also be delayed due to for instance lack of signaling channel resources for instance if many terminals generate ACK traffic that is so sparse that scheduling requests need to be generated with high frequency. A transport protocol design that tries to limit the amount of ACK traffic can have a performance benefit under such circumstances as the limitation is then more controlled and the protocol can be optimized for this. Reduced ACKs can unfortunately cause coalescing, something that may be mitigated to some extent by means of packet pacing.

3. 4G and 5G evolution

It is currently unclear in what aspect a 5G protocol stack will affect transport protocol performance. Listed below are however some features of evolved 4G and 5G that have relevance in this context:

o Shorter TTIs (Transmission Time Interval) has the potential to reduce the latency. Given that shorter TTIs have impact on the scheduling and also the retransmissions, the impact of shorter TTIs is that errors on the MAC layer will cause less jitter.

o Larger throughput variations can occur as a result of techniques like carrier aggregation and dual connectivity. Carrier aggregation means that additional carriers (possibly in very high frequency bands) are added. Dual connectivity can combine two similar or different radio access technologies on lower layers (below IP). Both technologies mentioned above can lead to large variations in available throughput.

o ECN is specified in 3GPP 36.300 [TS_36300]. ECN can provide with prompt indication of congestion without the need for packet drop caused by normal AQM operation, this can provide with a benefit for e.g. latency sensitive traffic. ECN also gives a explicit indication of congestion, opposed to the implicit congestion indications that loss and delay gives.

The shorter TTI feature is part of the 5G standardization, it should however be stated that the

4. Requirements for improved performance

The above considerations lead to a few things to consider when congestion control is designed for optimal performance in 4G and 5G networks:

- Avoid dependencies on precise RTT estimates: A typical real life case is the Hybrid Slowstart algorithm bundled with the Cubic congestion control. Uplink scheduling can break up transmission of ACKs which will in turn lead to increased RTTs that can falsely be interpreted as congestion.

- Use timestamps: Related to RTT estimate issue above. For instance a modified Hybrid Slowstart algorithm can take timestamp values into account and thus limit the effect of uplink scheduling effects that can distort the transmission of ACKs.

- Minimize latency under load: The quickly changing throughput in 4G/5G calls for a sensible balance between latency and throughput. Some amount of bufferbloat needs to be accepted in order to have enough data to utilize the radio resources optimally and get a high spectrum efficiency, this can however make the reaction to reduced throughput more sluggish. Hybrid loss/delay based congestion control can here be used to find a good balance between latency and throughput.

- ECN support: The transport protocols should support negotiation and use of ECN.

- Faster congestion window increase: Traditional AIMD (Additive Increase Multiplicative Decrease) based congestion avoidance algorithms are too slow to gain the benefits of e.g added carriers, therefore, more high speed alternatives should be considered, that are still reactive to congestion.

- Packet pacing: ACK compression effects can easily occur in 4G/5G networks, packet pacing should be encouraged to mitigate the coalescing effects caused by ACK compression, and will at the same time make ECN detection algorithms more robust.

- ACK reduction: Consider if it is possible to reduce the intensity of acknowledgements, especially in the uplink. Packet pacing may here be beneficial as it can mitigate the coalescing effects that can occur due to reduced ACK intensity.

5. Congestion control examples

This section lists a few examples of algorithm that can be useful

- o Default slowstart algorithms generally only operates at flow start-up and after a retransmission timeout. The drawback with this approach is that the congestion control cannot quickly grab new available capacity due to e.g the addition of an extra carrier. Fast Increase is a simple add-on to Cubic that resumes slowstart if it is detected that the bottleneck is underutilized for a while. A similar approach can be used for other congestion control algorithms.

- o Hybrid Cubic borrows the queue delay estimation from LEDBAT [RFC6817]. With this addition it is possible to set a target queuing delay that adds a limit to the congestion window based on network queuing delay in addition to the already existing loss based control of the congestion window.

- o Various High Speed congestion control algorithms such as SIAD (Scalable Increase Adaptive Decrease) [TCP_SIAD] can provide with improved performance in the presence of large changes in available throughput resulting from e.g added carriers.

The Fast Increase and Hybrid Cubic algorithms are described in more detail below. The code samples are shown with the Linux kernel 4.4 version of tcp_cubic.c as basis.

5.1. Fast Increase

The idea behind Fast Increase is simply to allow the Cubic congestion control to rapidly increase the congestion window if the RTT has been only marginally higher than the min RTT for a number of round trips, this is realised by forcing the ca->cnt variable to a low value. The HyStart delay algorithm used for this purpose. Code for this is shown below with the code from Linux tcp_cubic.c as basis

```
=======================
Function bictcp_acked(..) is modified according
to the code snippet below
New state variables are added, all initialized to 0
  u32 last_rtt_high
  u32 last_fast_increase
  u8 do_fast_increase
New constants
  #define N_RTT_LOW 5
  #define N_RTT_FAST_INCREASE 20
```

```
/** Old code **/
/* first time call or link delay decreases */
if (ca->delay_min == 0 || ca->delay_min > delay)
   ca->delay_min = delay;


/** New code **/
/*
 * Function bictcp_acked is modified to initiate fast increase when
 * RTT is lower than a given value for a given number of RTTs
 * srtt_l is a long term average of srtt
 */
if (ca->curr_rtt - ca->delay_min > (srtt_l-ca->delay_min)/2) {
   ca->last_rtt_high = bictcp_clock();
} else {
   u32 now = bictcp_clock();
   if (now - ca->last_rtt_high > N_RTT_LOW*ca->delay_min &&
       now - ca->last_fast_increase > N_RTT_FAST_INCREASE*ca->delay_min) {
     ca->do_fast_increase = 1;
     ca->last_fast_increase = now;
   }
}
/** End of new code **/


=======================
Function bictcp_update(...) is updated with a code snippet after
tcp_friendliness
/*
* Enable fast increase of the CWND
*/
if (ca->do_fast_increase)
  ca->cnt = min(ca->cnt,2);


=======================
Function bictcp_recalc_ssthresh(...) is modified
/*
* Resent do_fast_increase flag
*/
ca->do_fast_increase = 0;
```

Fast Increase code

ca->last_fast_increase should be updated to current time whenever a loss or ECN event is detected.

5.2. Hybrid Cubic

The Hybrid Cubic algorithm adds delay sensitivity to the Cubic congestion avoidance algorithm. It is, in the description below assumed that the timestamp option is enabled and that queue delay samples are computed, according to the description in LEDBAT [RFC6817]. Furthermore it is assumed than the timestamp clock frequency in sender and receiver are identical or that the sender can infer the timestamp clock frequency of the receiver and recompute timestamp values based on this information.

The function bictcp_update is updated according to the code snippet below.

```
New state variables
  float queue_delay
  u32 last_hycubic_cwnd_reduced
New constants
  #define QUEUE_DELAY_TARGET 0.1
  #define QUEUE_DELAY_GAIN_UP 100.0
  #define QUEUE_DELAY_GAIN_DOWN 0.2



/** New code, inserted before tcp_friendliness: **/
/*
 * The cnt variable is modified depending on the
 * relation between the OWD and the OWD target
 */
if (ca->owd < QUEUE_DELAY_TARGET) {
   float tmp = ca->queue_delay/QUEUE_DELAY_TARGET;
   int cnt_d = (int) (tmp*QUEUE_DELAY_GAIN_UP);
   /*
    * q is less than QUEUE_DELAY target
    * Increase cnt as OWD is approaching target
    * This will slow down congestion window growth
    * when owd increases
    */
   ca->cnt += cnt_d;
} else {
   /*
    * Set cnt to a high value,to prevent further growth
    */
   ca->cnt = 1000;
}
/** End of new code **/
```

```
/* TCP Friendly */
if (tcp_friendliness) {
   u32 scale = beta_scale;

   delta = (cwnd * scale) >> 3;
   /** New code **/
   if (ca->queue_delay < QUEUE_DELAY_TARGET) {
   /** End of new code **/
      while (ca->ack_cnt > delta) {  /* update tcp cwnd */
         ca->ack_cnt -= delta;
         ca->tcp_cwnd++;
      }
   /** New code **/
   } else {
      u32 now = bictcp_clock();
      if (now-ca->last_hycubic_cwnd_reduced > delay) {
         /* At most one reduction per RTT
         float overshoot = (ca->queue_delay-QUEUE_DELAY_TARGET)/delay;
         float alpha = min(0.5,overshoot*QUEUE_DELAY_GAIN_DOWN);
         ca->tcp_cwnd = (int)(ca->tcp_cwnd*(1.0-alpha));
         ca->ssthresh = ca->tcp_cwnd;
         ca->epoch_start = 0;
         ca->last_hycubic_cwnd_reduced = now;
      }

   }
   /** End of new code **/

   if (ca->tcp_cwnd > cwnd) { /* if bic is slower than tcp */
      delta = ca->tcp_cwnd - cwnd;
      max_cnt = cwnd / delta;
      if (ca->cnt > max_cnt)
         ca->cnt = max_cnt;
   }
}
```

Hybrid Cubic

Note that the code is not fully functional, for instance the floating point arithmetic need to be converted to fixed point ditto.

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

The possible outcome of this work has the same possible security considerations as other work around congestion control.

8. Acknowledgements

The following persons have contributed with comments and suggestions for improvements: Kristofer Sandlund, Mats Nordberg, Hans Hannu, Torsten Dudda and Szilveszter Nadas.

9. References

9.1. Normative References

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

9.2. Informative References

[LTE] "4G: LTE/LTE-Advanced for Mobile Broadband, Second Edition", 2013.

[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>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

[TCP_SIAD] "TCP SIAD: Congestion Control https://www.ietf.org/proceedings/91/slides/slides-91-iccrg-5.pdf", 2015.

[TS_36300] "3GPP TS 36.300", 2015.

Author's Address

Ingemar Johansson
Ericsson AB
Laboratoriegraend 11
Luleaa 977 53
Sweden

Phone: +46 730783289
Email: ingemar.s.johansson@ericsson.com