Self-clocked Rate Adaptation for Conversational Video in LTE

Ingemar Johansson
Ericsson AB
Laboratoriegränd 11
977 53, Luleå, Sweden
+46-1071 43042
ingemar.s.johansson@ericsson.com

ABSTRACT

This paper describes a rate adaptation framework for conversational video services. The solution conforms to the packet conservation principle and uses a hybrid loss and delay based congestion control algorithm. The framework is evaluated over both simulated bottleneck scenarios as well as in a LTE system simulator and is shown to achieve both low latency and high video throughput in these scenarios, something that improves the end user experience.

Categories and Subject Descriptors

C.2.6 [Computer Systems Organization]: Internetworking – Standards
Keywords LTE; Video; Conversational; Real time; self-clocking; WebRTC

1. INTRODUCTION

Conversational video is becoming more and more interesting; lately work has been initiated in the IETF and W3C to standardize conversational media for web-browsers (WebRTC). Rate adaptation is an important component as channel bandwidth is constantly varying. Wireless access such as LTE (Long Term Evolution) increases the importance of rate adaptation as the channel bandwidth of a default LTE bearer can change considerably in a very short timeframe. This means that a rate adaptation solution for conversational video in LTE must be both quick and also able to operate over a large span in available channel bandwidth. This paper describes a solution that borrows from the self-clocking principle of TCP and combines it with a new delay based rate adaptation algorithm, LEDBAT [1].

CSWS'14, August 18 2014, Chicago, IL, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2991-0/14/08…$15.00 .
http://dx.doi.org/10.1145/2630088.2631976

The paper describes the extra features needed to make the concept work well with conversational media.

2. PROPERTIES OF CONVERSATIONAL MEDIA

Conversational media typically constitute audio and video but may also involve a data channel for messaging and file transfer. The number of media streams may vary from a simple audio+video session to many audio and video streams and possibly also a set of data streams in a multiparty video conference. This paper will concentrate on the simple audio+video session.

3. PROPERTIES OF LTE CHANNELS

Wireless channels such as LTE can typically not guarantee a given bandwidth, this holds for true especially for default bearers. The network throughput may vary considerably in cases where the wireless terminal is moving around.

What happens if an application sends with a bitrate higher than the network can sustain is that packets are queued up. If AQM (Active Queue Management) is deployed, then packets are dropped if they are delayed for too long in the queues, first the packets are dropped gently but as the queuing delay becomes larger the packets will be dropped more frequently and possibly also in bursts, if AQMs are not used then the queuing delay may become very large, in the order of several seconds.

Excessive packet drops and/or high latency is something that should be avoided. In addition the grace time, i.e. allowed reaction time from the time that the congestion is detected until a reaction in terms of a rate reduction is effected, is generally very short, in the order of one RTT (Round Trip Time).

QoS (Quality of Service) enabled LTE bearers can be configured to prioritize for a minimum bitrate. This generally means that some headroom must be allocated for the case that the terminal moves into bad coverage. While it can be feasible to prioritize for a given bandwidth for 30kbps audio it is less likely that it can be done for 1500kbps video. The compromise here is to prioritize for a lower bitrate such as 200kbps. This is very helpful for a conversational video service as it improves the possibility to provide at least a minimum video quality even in highly congested situations. QoS is thus a good complement to rate adaptation.

4. REQUIREMENTS

The requirements on the rate adaptation framework can be summarized as: 1) Low latency and packet loss, the latency and packet loss rate must be kept as low as possible, even when the network throughput decreases. 2) Ability to compete with other traffic. 3) Stable bitrate, frequent fluctuations can be perceived as annoying.

As will be explained later, it can be difficult to combine requirement 3 with requirements 1 and 2 above. One could argue that a requirement "not generate too much congestion" should be added to the list of requirement. This is however already implied in requirement 1 as a solution that strives for low latency and packet loss will also react properly to signs of network congestion.

5. THE ADAPTATION FRAMEWORK

The adaptation framework has similarities to concepts like SST [2] Paceline [3] and TFWC [4]. One important property is self-clocking and compliance to the packet conservation principle. The packet conservation principle is described as an important key-factor behind the protection of networks from congestion [5]. An alternative described in [6] is rate based CC (congestion control) where the endpoint tries to estimate an optimal media bitrate based on packet reception statistics, further enhancements to this scheme are suggested in [7].

The packet conservation principle is realized by including a vector of the sequence numbers of received packets in the feedback from the receiver back to the sender, the sender keeps a list of transmitted packets and their respective sizes. This information is then used to determine how many bytes can be transmitted. A congestion window puts an upper limit on how many bytes can be in flight, i.e. transmitted but not yet acknowledged. The congestion window is determined in a way similar to LEDBAT [1]. This ensures that latency is kept low. The basic functionality is quite simple, there are however a few important steps to take to make the concept work with conversational media. These will be described in sections 5.1 to 5.3.

The congestion control feedback can be implemented either using RTCP feedback or with a shim layer between UDP and RTP as in the case of SST. The benefit with the latter is that it becomes more straightforward to add a data channel in the same framework. The benefit with RTCP is that it can be easier to standardize. RTP header extensions described in RFC5285 is yet another alternative. The remainder of this paper will focus on RTCP based congestion control.

The RTCP feedback is based on RFC4585 and is implemented as a transport layer feedback message, see example in Figure 1. The feedback control information part (FCI) consists of the following elements.

- Timestamp : A timestamp value indicating when the last packet was received which makes it possible to compute the one way (extra) delay (OWD).
- The ACK list, which makes it possible to detect lost packets and determine the number of bytes in flight. The ACK list is a possibly RLE coded sequence of the last K received RTP packets.
- ECN (Explicit Congestion Notification) echo, makes it possible to indicate if packets are ECN-CE (ECN-Congestion Experienced) marked.
- Source quench bit (Q), makes it possible to request the sender to reduce its congestion window. This is useful if WebRTC media is received from many hosts and it becomes necessary to balance the bitrates between the streams.

To make the feedback as frequent as possible, the feedback packets are transmitted as reduced size RTCP according to RFC5506.

```
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      |          length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  SSRC of packet sender                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  SSRC of media source                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Timestamp (32bits)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Last recv. sequence nr (16b)  | ECN echo      |Q|R|R|R|R|R|R|R|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    ACK vector (32b)                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1 Example RTCP feedback packet

The sender keeps a list of the 4-tuples {SSRC,TS_{Tx}, RTP_{SN}, RTP_{size}}. TS_{Tx} is the timestamp when the RTP packet is transmitted, RTP_{SN} is the sequence number and RTP_{size} is the size of the RTP packet. SSRC is also maintained, this makes it possible for the congestion control to handle several sources.

Based on each RTCP report a 3-tuple is formed {SSRC, TS_{Rx}, RTP_{SN}}. TS_{Rx} is the timestamp when the last RTP packet was received and RTP_{SN} is the sequence number of the last received RTP packet, SSRC indicates the source.

The number of bytes in flight is computed as the sum of bytes of all the RTP packets in the list of 4-tuples that have a sequence higher than the highest acknowledged sequence number. The one way extra delay (OWD) is computed according to the same principles as in LEDBAT [1]. The ACK list is used to compute the number of bytes in flight and to determine loss events. The ACK list can also be used to determine when to retransmit lost RTP packets, thus reducing the need for RFC4585 generic NACKs.

Figure 2 shows the functional overview of the adaptation framework. Each media type or source implements rate control and a queue (only one is shown in detail). The encoded frames are temporarily stored for transmission, RTP packets are picked from each queue based on some defined priority order or simply in a round robin fashion. In

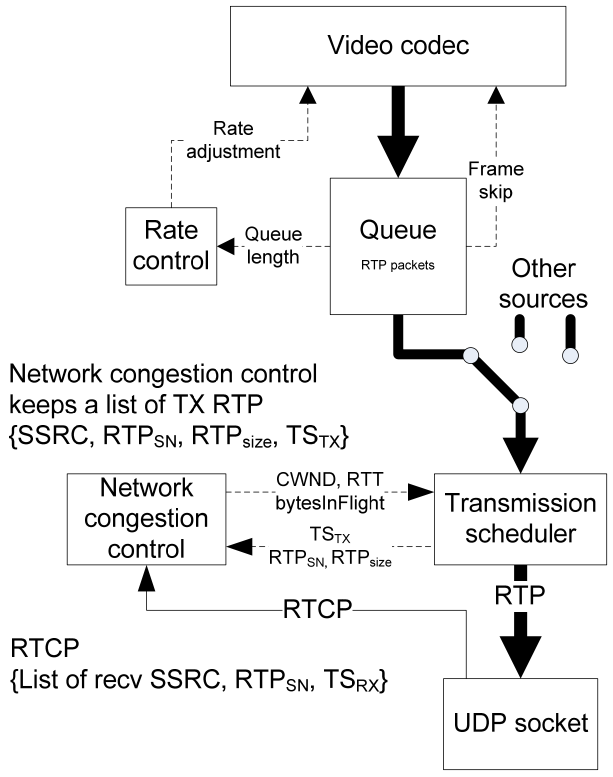
Video codec
Rate adjustment
Frame skip
Rate control
Queue length
Queue
RTP packets
Other sources
Network congestion control keeps a list of TX RTP {SSRC, RTP$_{SN}$, RTP$_{size}$, TS$_{TX}$}
Network congestion control
CWND, RTT bytesInFlight
TS$_{TX}$ RTP$_{SN}$, RTP$_{size}$
Transmission scheduler
RTP
RTCP
RTCP {List of recv SSRC, RTP$_{SN}$, TS$_{RX}$}
UDP socket

Figure 2 Overview of the Self-clocked rate adaptation framework

the general case all media must go through the transmission scheduler and is allowed to be transmitted only if the number of bytes in flight is less than the congestion window.

The solution to implement sending queues may seem awkward, the alternative would be to transmit frames directly as they are encoded. This approach is applied to the audio frames as they generally contribute very little to congestion due to the comparably low bitrate, moreover it keeps the latency for the audio at a minimum.

The problem with the above mentioned approach applied to large video packets is that as soon as the frames are transmitted, they are out of control for the sender and the frames instead risk being queued up in the network. The idea with sender queues and the self-clocking implemented here is to minimize the risk of excessive queue buildup in the network; as a consequence, encoded video frames may occasionally be queued up on the sending side, the consequence is that video will lag behind the audio. It must however be understood that the sender queues are most often empty and that they function as shock absorbers that avoid excessive delay or packet loss in the network, something that would affect the audio too.

The rate adaptation solution constitutes three parts; congestion control, transmission scheduling and media rate adaptation.

5.1 Congestion control

The congestion control sets an upper limit on how much data can be in the network (bytes in flight); this limit is called CWND (congestion window) and is used in the transmission scheduling.

A congestion control method, similar to LEDBAT, measures the OWD. The congestion window is allowed to increase if the OWD is below a predefined target, otherwise the congestion window decreases. The delay target is typically set to 50-100ms. This ensures that the OWD is kept low on the average. The reaction to packet losses is similar to that of loss based TCP, i.e. an instant reduction of CWND. LEDBAT is designed with file transfers as main use case which means that the algorithm must be modified somewhat to work with rate-limited sources such as video. The modifications are:

- Allow the congestion window to increase even though it is not fully utilized. The solution is to allow the congestion window to increase as long as the number of bytes in flight is greater than a fraction of the congestion window. This is especially important for VBR (Variable Bit Rate) video as the temporal bitrate may vary very much and too restrictive rules around the increase of the congestion window will only slow down the ramp-up of the media bitrate. One can also say that this relaxed rule compensates for the source limitation and helps to avoid starvation caused by competing flows such as file transfers. The idea behind this is inspired by the work on TCP congestion window validation [8].
- Avoid that the congestion window is left at high levels in cases where the source bitrate becomes low, for instance when the video stream is put on hold or when the video bitrate has decreased for other reasons. The method is to gradually decrease the congestion window if the number of bytes in flight is lower than a fraction of the congestion window for a prolonged period. This solution reduces the risk of excessive delay or packet loss when video bitrate increases. A similar method is proposed for TCP [8].
- Fast start makes the video bitrate ramp-up within 3 to 5 seconds. It consists of two components, one slow-start component similar to TCP in the network congestion control part which exits when OWD is greater than half the delay target, and one component that makes the video bitrate increase quicker in the initial phase.
- Make the delay target adaptive. The delay target is increased and eventually limited by a maximum level

when competing flows are detected. The effect is that the congestion control shifts from being delay sensitive to being loss sensitive. The benefit with this feature is when the rate adaptation algorithm has to compete with e.g. file transfers, which typically use TCP algorithms that are only sensitive to loss. Frame blanking is used to mitigate the issue with self-inflicted congestion, the technique is to stop sending video or other high bitrate media for a short period (roughly one RTT) and measure the OWD during this period. The delay target is then adjusted based on the OWD during the blanking period. Figure 3 shows an example where a video and a 500kByte FTP transfer compete for bandwidth over a 1Mbps bottleneck. The FTP transfer starts at T=16 and completes at T=23. The delay target is adjusted up as a consequence of the increased OWD. The delay target continuously decays by a small amount, video is blanked every 10s to make this decay quicker. A video blanking event is present at T=26.5s, this makes the delay target to drop as no competing flow is detected.

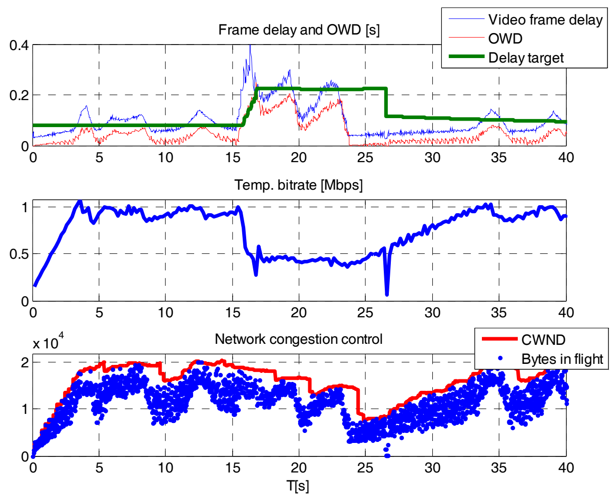

Figure 3 Example with adjustable delay target

5.2 Transmission scheduling

Transmission scheduling limits the output of data, given by the relation between the number of bytes in flight and the congestion window similar to TCP. Two additions deserve mention:

- Self-clocking algorithms like TCP and also the rate adaptation algorithm described here shows a tendency to cause packets to coalesce, i.e. packets become transmitted in bursts. This can cause unnecessary packet drops, increased jitter and also a more unstable behavior in packet schedulers in the radio. The remedy is to add a timer that triggers packets to be transmitted at a speed that is not higher than the estimated throughput. This is known as packet pacing. The timer interval is determined by the CWND and the estimated RTT. This method efficiently reduces issues with coalescing.
- Allow bytes in flight to exceed CWND when OWD is low. The number of extra bytes that the CWND is allowed to be exceeded is determined from the relation between the OWD and the delay target. This improves performance especially when VBR video is used and also when the video encoder transmits occasional I-frames. The feature breaks the packet conservation principle somewhat and can potentially cause delay spikes or more packet loss if the transmission path is close to being congested.

5.3 Media rate control

The media rate control serves to adjust the media bitrate to ramp up quickly enough to get a fair share of the system resources when link throughput increases.

The reaction to reduced throughput must be prompt in order to avoid getting too much data queued up in the sender frame queues. The queuing delay is determined and the media bitrate is decreased if it exceeds a threshold. The amount by which the media rate should be decreased can be determined from the frame queue delay. The media bitrate can also be directly decreased because of indications of packet loss, ECN-CE and detection that the OWD exceeds the delay target, the latter can however have negative implications for instance if the scheduling jitter, which may not be congestion related, is high.

In cases where the sender frame queues increase rapidly such as the case of a RAT (Radio Access Type) handover it may be necessary to implement additional actions to ensure that the sender frame queues are drained quickly, examples are: 1) Drop encoded frames: The encoder can be notified about the dropped frames in the frame queue and can take corrective actions to repair codec state. 2) Skip frames before encoding: In other words reduce the frame rate. This can be a good remedy in cases where it may take time to e.g. change encoder setup. Of the two methods above, the method to skip frames before encoding is to prefer as the codec state is preserved but there could be cases where it becomes necessary to drop already encoded frames, for instance in cases when the path throughput drops dramatically or even drops to zero.

6. SIMULATIONS

A state of the art dynamic LTE simulation with settings according to Table 1 was used to assess the performance of the algorithm. The simulator models the whole protocol chain including handover, radio propagation etc. Figure 4 shows an example trace for an individual user. A more severe congestion period occurs at T~15.6s where it is visible that the acknowledgements (Bytes newly ACKed) are more sparse. As a consequence of this, the transmission of video frames is delayed already at the sender. The video encoder senses that the sender queue is building up and reduces the video bitrate. Also is seen that the OWD is increasing and as a consequence, the congestion window is also reduced. The transmission of audio frames is not delayed at the sender which means that the audio frame delay does not go above ~130ms while the video frame delay is higher. The difference between the video and audio frame delay indicates the amount of queuing in the sender, it is clear that the sender queue is basically empty, except for a short period between T=15.7s and T = 16.1s

Table 1 Simulation settings

Variables	Value
Cellular layout	21 cells (7 sites); 3GPP case 1 settings
System setup	10MHz bandwidth, 2GHz carrier frequency, eNB transmission power 40W, and MIMO transmission mode
Channel	Typical Urban
Propagation model	Okumura-Hata model
Scheduler	DL scheduler: Proportional fair UL scheduler: Proportional fair
User generation	Poisson arrival based user generation
Mobility	UE moves straight in a randomly selected direction, at a speed 3km/h, ideal handover model (no user plane interruption, no handover signaling)
Traffic scenario	Video: Rate adaptive video, codec : H.264, bitrate 150-1500kbps.RTCP BW: 75kbps Audio: Frame size 20ms, bitrate 20kbps, No silence suppression. FTP: fixed number of FTP users corresponding to an average of 4Mbps load per cell, NewReno TCP.

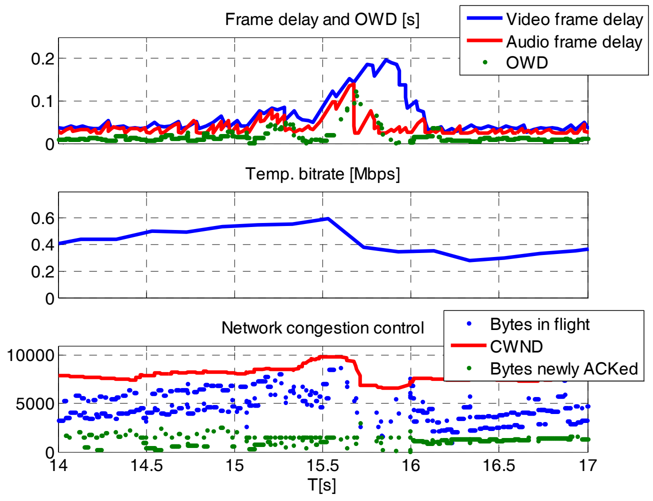

Figure 4 Upper: Video and audio frame delay. Mid: Temporal video bitrate. Lower: CWND and bytes in flight

One may argue that fluctuations in bitrate should be avoided, this is generally true and one could imagine a solution that tries to settle for a lower but more stable bitrate. The problem is that, in order for an application to learn the capacity of a transmission path, it is also necessary to probe the path. A solution that tries to find a lower but more stable bitrate will have difficulties finding that bitrate as there are currently no reliable signals available from the network that informs about that, for instance the delay increases only when the bitrate is getting close to the limit, likewise congestion related packet drops also only occur close to the limit.

The self-clocked algorithm is compared against the rate based CC algorithm in [6]. The load level i.e. the intensity of new connections, is varied from 2 to 16 video users/cell. The tail latency (Figure 5) indicates how large the delay spikes are. It is clear that, while the self-clocking rate adaptation manages high load levels quite well, the rate based CC cannot handle increasing load. Figure 6 shows that, even though rate based CC generally gives a lower average bitrate, it seems like the reaction to congestion is either not sufficient or not timely enough to avoid delay spikes. The likely reason to the low throughput for the rate based algorithm at low loads is not studied in detail. The conclusion is nevertheless that the self-clocked rate adaptation algorithm presented here performs better than the rate based CC algorithm

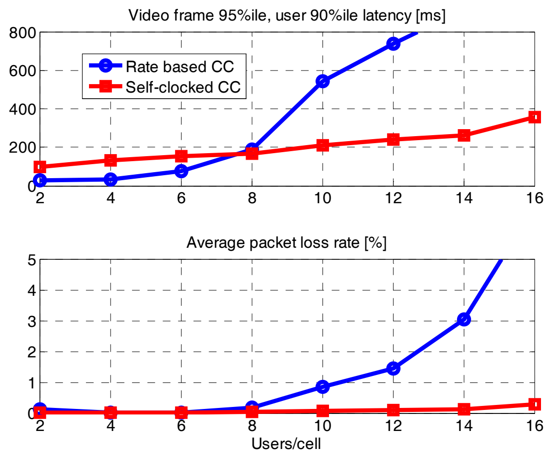

Figure 5 Tail latency.

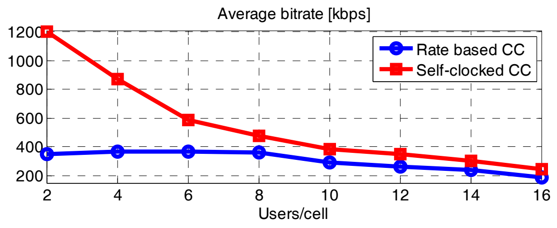

Figure 6 Average video bitrate

7. DISCUSSION

The results presented in section 6 are promising and clearly show that it is possible to operate conversational video services in LTE networks. The presented framework also performs better than the rate based CC described in [6]. The reason to the different performance can be found in how the two alternatives operate. The self-clocked congestion control has several complementing control mechanisms: 1) Increased delay makes the congestion window decrease, but the media bitrate is not immediately decreased. 2) Delayed acknowledgements slow down packet transmission as the number of bytes in flight is not allowed to reach above the congestion window, the media bitrate is however not immediately decreased. 3) Increased sender queue eventually lead to a decreased media bitrate. The rate based congestion control has a single control mechanism in that an increased delay leads to immediate reduction in media bitrate. The multiple, complementing mechanisms in the self-clocked congestion control makes it possible to reach both high link utilization and a prompt reaction to congestion. The rate based congestion control however has only one mechanism that needs to strike a balance between sensitivity to jitter that is not necessarily congestion related and prompt reaction to congestion (within one RTT). Moreover, the sensitivity to lost or delayed feedback information is smaller for the self-clocked congestion control as the packet transmission is slowed down in such cases.

Despite the promising results with the self-clocked rate adaptation algorithm, there are a few concerns, many of these are shared with other rate control algorithms such as rate based CC.

- Over-reaction at handover: The outlined algorithm cannot easily distinguish between congestion and the possible delay spike that may occur when packet transmission is temporarily put on hold during handover. This leads to unnecessary rate reductions at handover and can be a problem if the mobility is high.
- Over-reaction to congestion events: Endpoint rate adaptation algorithms do not know the exact capacity of the transmission path, which means that it is safer to reduce the bitrate a little extra when congestion occurs. This easily leads to a too large rate reduction.
- Impact from other traffic: Other, potentially aggressive cross-traffic can cause degraded performance for the delay and loss sensitive conversational video, rate adaptation for the latter helps very little in this case.
- Frequent and large media bitrate changes: The media bitrate must be reduced promptly when congestion is detected, in addition the bitrate change may be substantial. A user may perceive this as annoying.
- VBR encoded video: The framework has a few features for improved handling of variable bitrate encoded video. These features however increase the risk of delay spikes and/or packet loss. It is therefore recommended to make the video bitrate as smooth as possible.

From the network side, there are a number of features that can improve performance: 1) The EPS (Evolved Packet System) specifies several QoS features. Especially for enterprise solutions, QoS is a very important complement and it can limit the negative impact of aggressive cross-traffic. Experiments have also shown that the self-clocked rate adaptation algorithm benefits from a QoS enabled prioritized minimum bitrate. 2) ECN is a complement that can be helpful as it clearly indicates congestion opposed to the implicit means to detect congestion (loss, delay).

8. CONCLUSION

This paper shows that it is possible to implement well functional conversational video over normal LTE default bearers and get very good performance even at high load levels. The quickly changing channel characteristics however put some requirements on the solution. A solution that conforms to the packet conservation principle and leverages on novel congestion control algorithms and recent TCP research, together with media bitrate determined by sender queuing delay and given delay thresholds, is shown to meet the goals of high link utilization and prompt reaction to congestion. The solution is realized using feedback over RTCP but can also be implemented by using a shim layer between RTP and UDP or with RTP header extensions.

9. ACKNOWLEDGEMENTS

The author wish to thank the following persons for valuable feedback during this work: Bo Burman, Laurits Hamm, Hans Hannu, Nikolas Hermanns, Mats Nordberg, Jonas Pettersson, Kristofer Sandlund, Zaheduzzaman Sarker, Per Synnergren, Ylva Timner, Min W Wang. Gorry Fairhurst was helpful during the final review of this paper.

10. REFERENCES

[1] IETF RFC6817 Low Extra Delay Background Transport (LEDBAT)

[2] Bryan Ford, “Structured Streams: a New Transport Abstraction”, ACM SIGCOMM 2007

[3] Erbad et. al. “Paceline: Latency Management through Adaptive Output”, ACM Multimedia Systems Conference 2010

[4] Choi, Handley “Fairer TCP-Friendly Congestion Control Protocol for Multimedia Streaming Applications”, CoNEXT’07, December 10-13, 2007

[5] Mathis et. al. “Forward Acknowledgement: Refining TCP Congestion Control”, ACM SIGCOMM 2006

[6] Alvestrand et. al “A Google Congestion Control Algorithm for Real-Time Communication”, work in progress, http://tools.ietf.org/id/draft-alvestrand-rmcat-congestion-02.txt

[7] Mascolo et. al “Understanding the Dynamic Behaviour of the Google Congestion Control for RTCWeb” Packet Video Workshop (PV), 2013 20th International. IEEE, 2013

[8] Fairhurst et. al. “Updating TCP to support Rate-Limited Traffic”, work in progress, http://tools.ietf.org/wg/tcpm/draft-ietf-tcpm-newcwv