Media Rate Adaptation for Conversational Video Services in Next Generation Mobile Networks

Medienratenanpassung für Videokommunikationsdienste in mobilen Netzwerken der nächsten Generation

Master Thesis

at

Lehrstuhl und Institut für Nachrichtentechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen

Univ.-Prof. Dr.-Ing. Jens-Rainer Ohm

by

Nikolas Hermanns
Matr.-Num. 281526

Supervisors:
Dipl.-Ing. Laurits Hamm (Ericsson GmbH)
Dipl.-Ing. Christian Feldmann (RWTH Aachen University)

17. December 2013

Acknowledgements

This thesis has greatly benefited from the support of many people, some of whom I would sincerely like to thank here.

To begin with, I am deeply grateful to Professor Jens-Rainer Ohm and Laurits Hamm for offering me such an interesting topic of investigation.

Furthermore, I have greatly appreciated the suggestions of Christian Feldman, Ingemar Johansson, Thorsten Lohmar, Zaheduzzaman Sarker, Markus Andersson and Victoria Matute Arribas.

My thanks also go to Stephanie Müller and Fiona Williams for the great and helpful feedback on this work.

On a more personal note, I would like to thank my parents for encouraging and supporting me.

Declaration of Authorship

I hereby declare and confirm that this is entirely the result of my own work. I have only used the stated documents and aids, and also indicated citations correctly.

_____ Aachen, 17. December 2013

Table of Content

Chapter 1 Introduction

Many operators around the world have been launching 4G LTE (Long Term Evolution) mobile networks during the last years. Currently, LTE is mostly used to provide high speed mobile data. Telephony services via LTE (VoLTE) can be offered using IP multimedia subsystem (IMS) and Multimedia Telephony services. Today, VoLTE deployments for voice are in operation and VoLTE video will be the next step to offer video telephony via LTE and IMS [1]. In addition, many Over-The-Top video conversation services are performed in mobile networks today and their number is growing. Increasing the efficiency of the transmission using LTE is now in focus in standardization groups, such as the Internet Expert Task Force (IETF).

“RTP Media Congestion Avoidance Techniques” (RMCAT) is a working group of IETF, started in 2012. The group is currently active and Ericsson Research is a member. It aims to specify one or more, generally acceptable, media rate adaptation frameworks for congestion avoidance. A further description for the reason of media rate adaptation algorithm is given in Chapter 2. This thesis analyses how the adaptation algorithms behave in a realistic LTE multi user scenario. A good adaptation framework for cellular wireless networks must overcome the following challenges [2], [3]:

- It must cope with dramatic temporal variations in link rates.
- It must avoid over-buffering and high end-to-end delays, but at the same time, if the media rate can be increased, it must avoid under-utilization.
- It must be able to handle outages without over-buffering, cope with asymmetric outages, and recover gracefully afterwards.
- It must cope with different media behavior, i.e., the media should contain idle and data-limited periods. An example is video, which is highly dependent on the encoded content (amount of motion, details of the scene).
- It must work in competition with session, which have the same rate adaptation, and handle competing sessions with different adaptation control.

The motivation of this work is to contribute to the work of RMCAT by presenting a concrete evaluation framework. Furthermore, it aims to evaluate three of the latest media rate adaptation algorithms for real-time communication in the 4G LTE. To two of them, the RMCAT work group drew attention. The third algorithm is from Ericsson Research and is yet under development. Weaknesses of the algorithms are shown, solutions and hints how to cope with these are given. Scenarios were chosen for the evaluations so as to demonstrate how the three algorithms behave in relation to the set of challenges described above.

1.1 Highlights of this Work

There are three outstanding features of this work. Firstly, a detailed mathematical construct is presented in Chapter 3 to evaluate the algorithms on a common base. Therefore, key performance indicators, which are commonly used, are extracted from the literature and merged into a mathematical concept to fit together.

The second highlight is the usage of a very detailed LTE system simulator, which is further described in Chapter 5. It simulates multiple users in a complete LTE system. The advantage towards traced based simulations, which are commonly used, is that a conclusion to the interaction of the users can be given. In mobile radio networks the characteristic of the allocated radio resources is highly depending on the client behavior. So, if a client does not use the radio resources they will not be allocated to it. Therefore, the maximal throughput for the user depends on the behavior of all other users. Traced based simulations measure link characteristics in advance. Since the algorithm taken to measure these traces is not the algorithm which is evaluated, the conclusion of this simulation is arguable. A multi user scenario shows the influence of the algorithm itself and of other session types.

The third highlight of this thesis is that as a first work a comparison of the three algorithms can be given. Although the investigated algorithms are developed for the same purpose conceptual and thus performance differences are shown. The thesis shows also the influence between the algorithms, which is a big step forward to a realistic statement, since the types of algorithms which are used for rate adaptation in the future will for sure be varying.

Beside these outstanding features, this thesis discovers problematic weaknesses in all three algorithms and gives hints to eliminate these weaknesses. Concerning the Sprout algorithm, which is one of the investigated algorithms, the thesis proposes a solution to cope with conversational video, called E-Sprout.

As a novelty this thesis presents a Monte Carlo simulation to evaluate the algorithms in wide range of network conditions. Thus, a new way to discover weaknesses of media rate adaptation algorithm is shown.

1.2 Structure of this Thesis

This report begins with a description of the difficulties of supporting media rate adaption for conversational video in Chapter 2. Chapter 3 presents the detailed mathematical construct for the evaluation. In order to provide the reader with a deep understanding of the investigated algorithms, a detailed theoretical explanation of each of the investigated algorithms is provided in Chapter 4. In Chapter 5 the structure of the evaluation framework, and the scenarios simulated, are presented. This evaluation framework is used to produce the simulations results on the performance of the algorithms, elaborated in Chapter 6. This Chapter includes an overview of the advantages and weaknesses of each algorithm, and additionally, a detailed presentation of the features of the algorithms. Finally, Chapter 7 discusses the overall results and gives a conclusions of the work.

1.3 Related Work

[2] gives a detailed explanation of the concept of the Sprout algorithm. The implementation from Sprout is taken from http://alfalfa.mit.edu/. [4] describes the concept of the Google Congestion Control algorithm and the implementation from this algorithm is taken from http://webrtc.googlecode.com. The Self-Clocked algorithm is an Ericsson internal proposal.

In [5] the metrics for the evaluation of congestion control mechanisms are named. This document is a guideline which gives an overview of the key performance indicators and where to find them.

In [6] the congestion control requirements of the RMCAT workgroup are brought together. This overview gives a detailed introduction about the requirement of a media rate adaptation framework.

From [3] the setup for the evaluation is taken. This paper suggests under which circumstances a media rate adaptation framework for real-time conversation should be tested.

Chapter 2 Media Rate Adaptation for Conversational Video

As this thesis focusses on video telephony via LTE this chapter aims to clarify the problems which come along with telephony in mobile networks and specify a solution for these problems. The media bitrate for video is commonly many times higher than for audio. If a bitrate of 20 kbps for AMR-WB audio and a bitrate of 400 kbps for H.264 standard definition video is assumed, the required bitrate for video will be 20 times higher than for audio. The majority of mobile devices supporting LTE also supports high definition (HD) video. For HD video at a good quality one should assume a minimum bitrate of 1 Mbps and higher. Next generation video codecs like H.265 (HEVC) are expected to achieve the same video quality at almost half of the bitrate. Still, the required bitrates for video will be high and more efficient video codecs will not only be used to reduce bitrates but also to increase video resolution and quality.

If a multitude of users, which are allocated in an LTE cell at the same time, share the same radio resources and use up the available capacity, the result is a changing available bitrate per user over time. Mobility of users, radio propagation and shadowing are other source of available bitrate variations. Although video sessions easily use up the available bandwidth in LTE, a realistic scenario also concerns different types of services, e.g., file transfer, web browsing and streaming. This puts another effort to provide all this services at once, since the characteristics of these services are different.

If the overall load in a radio cell increases and reaches the cell capacity, congestion will occur and timely transmission of data packets is not possible. It is the task of the radio scheduler to allocate radio resources to the users according to their service profile. If a prompt transmission of packets is not possible, packets can either be delayed or dropped. In packet based transmission the packets are usually queued, if they cannot be transmitted at the current time. Queues can then be emptied at a later point. This method has the disadvantage that the queuing of packets causes end-to-end delays and if the maximum size of the queues is reached, packets will even be dropped. The opportunity, that the network provides a guaranteed bitrate is further discussed in Section 5.1, but not applied in this thesis.

Real-time conversational services are very delay critical, meaning that the user experience will degrade largely if the end-to-end delay or one way delay (OWD) increases too much. Acceptable OWDs for voice telephony are around 200 ms [7] (mouth to ear), while the acceptable OWDs for video telephony are around 400 ms [8]. Nevertheless, this delay puts a very strong requirement on the transmission for video, as a video stream is more likely to experience congestion than voice telephony due to the higher video bitrate.

In order to cope with that problem media rate adaptation algorithms are used. They are described as frameworks to fulfil the following tasks:

- The algorithms should prevent queuing in the network avoiding too long OWDs and packet losses. This is done in the same way as congestion control algorithms avoid congestion. However, the requirements for interactive, point-to-point real time multimedia, which needs low-OWD and semi-reliable data delivery, are different from the requirements for bulk transfer like FTP or bursty transfers like Web pages. For example TCP sessions use packet loss as a metric to measure congestion. For real-time communication the occurrence of packet loss should be prevented completely [6].
- The algorithms have to control the encoder to adapt the video bitrate according to the available bitrate of the channel. If the available bitrate increases the video bitrate can be increased to send the video at higher quality. If the available bitrate decreases, the video bitrate should also decrease to avoid a huge amount of buffered data and simultaneously a growing OWD.

Thus, media rate adaptation tries to serve best possible video quality and avoid too long OWD and packet loss.

Chapter 3 Key Performance Indicators

This chapter describe Key performance indicators (KPI), which are commonly used, extracted from the literature and merged into a mathematical concept to fit together. In the term of algorithm describe specific characteristics of the algorithm. Channel metrics are measured and then evaluated to produce the KPIs. The main purpose of these indicators is to provide a comparison independent from design and technologies. Common indicators, have to be considered.

Different KPIs sometimes show correlation, and are not necessarily independent from each other. In some cases they also influence each other and statements have to be set in comparison. Some of these KPIs are also used to grade the algorithm at run-time. In order to define the key performance indicators the channel metrics need to be defined:

The throughput x [5] is the number of bytes sent by the sender over time. It can be defined as follows:

$$x(i) = \frac{1}{T} \sum_{t=(i-1)*T}^{i*T} length\big(ftx(t)\big) \tag{1}$$

ftx is the vector of sent packets and i is the evaluation step, which is periodic with T the evaluation sampling rate. i is defined as $i = \left\lfloor \frac{t}{T} \right\rfloor$.

The frame loss rate L indicates the number of bytes lost in the interval T. The frame loss rate can be described as follows:

$$L(i) = \frac{1}{T} \sum_{t=i*T-T}^{i*T} length\big(fl(t)\big) \tag{2}$$

fl is the vector of lost frames.

In order to get an overview of all relevant KPIs a division of the indicators is helpful. In the case of media rate adaption algorithms the indicators can be divided into three types:

End-to-End KPIs: Describe the user experience and evaluate the end to end media measurements.

Algorithm KPIs: Evaluate the adaptation algorithm, how fast, stable, exact and reliable the algorithm performs.

System KPIs: Are used to show the benefit of the algorithm concerning system performance. They evaluate how efficient the system resources are used and how fair the resources are shared between all users and services.

In the next three subchapters the different KPIs types will be described in more detail.

3.1 End-to-End KPIs

For the user, a smooth and homogenous experience is most important. The factors, which influence the quality of a video call the most, are the end to end delay and the video quality. Besides achieving a high video quality an even higher attention has to be given to the constancy of the quality.

3.1.1 Video Bitrate and Frame Loss Rate

To measure the quality of the video indicators can be combined. The video bitrate or encoder bitrate, and the frame loss rate give a first impression of the video quality. The video bitrate is number of bytes received by the decoder. It is a subset of the throughput. It excludes frames which are dropped anywhere in the channel, fragmented frames which cannot be fixed and none video data like headers etc. The maximum video bitrate is A the available bitrate of the channel.

The frame loss rate is calculated with

$$FLR = \frac{L(i)}{x(i)}. \tag{3}$$

Here $L(i)$ is the frame loss rate and $x(i)$ is the throughput rate both at step i. In words FLR stands for the rate between all bits which where send form the sender and all corrupted or lost bits. Although [9] pronounces $L(i)$ should include frames which are not received in time to be used for decoding, in this evaluation the frames which are not received in time are not considered as frame loss. These delayed frames are only counted in the OWD. The advantage of this is a traceable packet loss for each algorithm and clear separation of packet loss and OWD. If a frame can't be decoded because some bits are missing the whole frame is counted as a lost frame.

FLR can be divided into two considerations, the FLR over a short term and the FLR over a long term. A short high amount of frame losses cannot be eliminated in some cases but this is not very problematic. The video will be scratched for a short time and after the next I-Frame it will be intact again. In long term a high amount of frame losses is destructive for the video stream, but a little amount of packet losses could be concealed by the decoder without a noticeable quality reduction in the reconstructed video.

Both indicators, video bitrate and FLR in the case of end-to-end KPIs are considered as endpoint indicators, per user.

3.1.2 One Way Delay

The OWD, which is also known as the end-to-end delay, is measured at the receiving side. This delay is measured from camera to screen which means to measure the time the video is actually recorded until it is presented on the video screen at the receiver. In order to

measure this time the receiver and the sender have to be synchronized in time. A simulator uses the system time to achieve this.

In the case of OWD constancy is not so important for the user. There is no drawback, for the user, if this delay changes frequently over time. However, it should be avoided that the OWD exceeds its upper limit. If the OWD is above its limit and the frame cannot be decoded in time, this frame can be considered as a lost frame.

In [7] it is said, that the OWD should be kept well within the acceptable levels of 150 ms, and must not exceed 400 ms. The guideline in [8] also recommends to meet 100 ms and 150 ms. In [10] and [8] it is said that a OWD of 350 ms is still acceptable and is produced by commercial videoconferencing systems. For this evaluation the maximum OWD is set to 400 ms.

3.2 Algorithm KPIs

The algorithm tries to adapt the encoder bitrate to the available bitrate. The KPIs presented above are also valid for the algorithm. A strict division of end-to-end KPIs and algorithm KPIs cannot be made.

3.2.1 Oscillation and Stability

The minimization of oscillations of OWD and throughput is an important KPI in case of algorithm KPIs. Although the oscillation of OWD is not relevant for the user, it can be destructive for an algorithm. Stability is associated with rate fluctuations or variations [5]. To measure rate variations the standard deviations σ of per-session throughput [11] or the coefficient of variations CoV [12] can be used. Let's assume that $\bar{x}$ is the average throughput in the interval [1,m] and is defined as

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x(i) \tag{4}$$

with $x(i)$ the throughput at step i. The standard deviations of per-session throughput in the same interval is

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x(i) - \bar{x})^2}. \tag{5}$$

The coefficient of variation is

$$CoV = \frac{\sigma}{\bar{x}}. \tag{6}$$

In [13] the coefficient of variation is named stability index. The smaller the stability index is, the less oscillation a source experiences. The frame loss rate is related to the stability of the OWD and throughput and will be minimized if the algorithm is more stable [5]. [3] says that in

order to reach a favorable equilibrium, it is important that the algorithm is balanced between stability and reactivity. If it is too instable, small variation in cross traffic will be amplified which can lead to high OWD and packet loss. If the algorithm is too stable or too conservative, it will not reach the available bitrate and cannot adapt if congestion occurs.

3.2.2 Total Adaptation Response Time, Responsiveness and Reactivity to Events

The stability of an algorithm is bonded to the total adaptation response time also referred to as the responsiveness. If an algorithm can provide a short total adaptation response time for every step of adaptation, the accuracy of the adaptation will increases but the stability of the algorithm will decrease. Figure 1 shows the total adaptation response time in the step interval [1,m]. This time is widely changeable through the adaptation algorithm.

Although the minimum of this time is system and condition specific, the growth of this time is influenced by the algorithm and is called *self-inflicted (queuing) delay*. The *detection time* is an algorithm specific parameter as well as the *time to next adaptation request*. To send this request back to the sender the network needs a specific amount of time but the *self-inflicted queuing delay* is added to this time. Cellular carrier provisions generally separate uplink and downlink queues for each device in a cell. So, the number of packets in the queue and the resulting delay are self-inflicted [2]. The time the network needs to clear the queues depends on the number of packets in these queues. Although algorithms, which work in conversational video services, should deliver an acceptable throughput, another important purpose is to lower that number of packets in the network. If the algorithm is not filling the queues with the right amount of packets it will not reach the available bitrate of the channel.

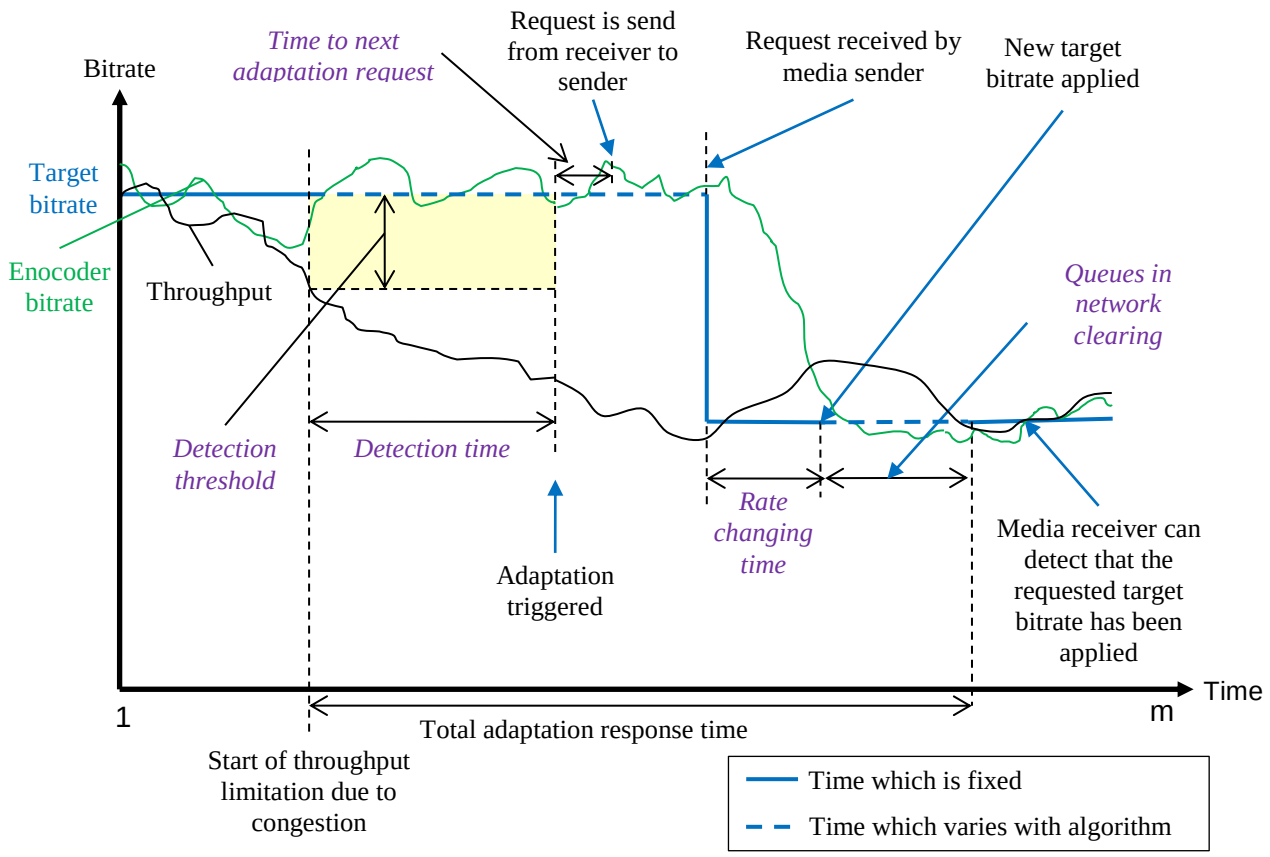

Figure 1 Total adaptation response time.

To form a KPI out of this the total adaption response time can be expressed as the responsiveness. It is defined as the time it takes for the cumulative average throughput to reach $\pm 10\%$ range of the average throughput defined in equation (4) which is measured from one occurrence in change of the channel to the next [11]. This step is i_{change}. The cumulative average throughput at step i is the average throughput from step i_{change} up to step i, while $i_{change} \leq i \leq m$ and m is the last step before the next change of the channel will occur. Equation (7) expresses the cumulative average throughput where a indicates each occurrence of change of the channel [13].

$$\tilde{x}_a(i) = \frac{1}{i} \sum_{k=1}^{i} x(k) \tag{7}$$

The responsiveness is now defined as

$$R_a = \frac{\max\left\{k: \left|\frac{\tilde{x}_a(k) - \bar{x}}{\bar{x}}\right| > 0.1\right\}}{m}. \tag{8}$$

In words, the responsiveness defines the ability of the algorithm to fulfill a rate adaptation before the next change of the channel will occur.

Another factor limiting the stability is its reactivity to events. If an algorithm is more insensitive for changes of the available bitrate, it will behave more stable, but unfortunately it will deliver the wrong estimated bitrate. On the one hand it should be reactive to changes which are consistent and neglect changes, which will vanish after a short time, on the other hand if consistent changes are rapid it should follow them fast. In [5] the short time of the neglected changes is defined as the connection's round-trip time RTT.

3.2.3 Aggressiveness

In [12] and [14] the aggressiveness is defined as the maximum increase in sending rate in one round-trip time RTT, in bytes per second, given the absence of congestion. Let us assume a window m, which has the length of the connections RTT, than

$$Aggr = \max_{i,j}\{x(j) - x(i)\} \tag{9}$$

is the aggressiveness of the algorithm with $|i - j| \leq m$ and $i < j$. In words, The aggressiveness is the fast acceleration of an algorithms sending rate to improve network utilization when there is a possible increase of sending bitrate to the available bitrate.

3.2.4 Smoothness

In [12] and [14] the smoothness is defined as the variation of the sending rate in one RTT in a deterministic steady-state scenario, which means a scenario where any algorithm should stabilize in bitrate and delay to a certain and equal amount. The mathematic formula is the same as for CoV defined in equation (6). The difference to the CoV is the setup of the evaluation. The CoV is not calculated in a steady-state scenario.

3.2.5 Overhead

In battery-powered devices the energy consumption of the end-to-end flow is one of the key interests. There are two ways to compare different algorithms in case of energy consumption. Firstly, the overall energy consumption can be computed. However this has to be set in line to the actual achieved video bitrate VB. An algorithm which achieves higher video bitrate has also a higher energy consumption than one which achieves lower. However, a high video bitrate is strived for in order to fulfill the user's demands. So secondly, a more precise measurement would only consider the overhead in sending rate which the algorithm needs. This overhead is equal to $x(i) - VB(i)$ and it is the overhead from all protocol layers with the additional overhead of the algorithm, including the feedback. In a generalized scenario it is unnecessary to estimate the real energy consumption which is associated to the overhead because it is enough to calculate the ratio between overhead and video bitrate, like [15] says. To bring this into an KPI which is bounded between 0 and 1, the overhead rate in the interval [1,m] can be described as follows:

$$Or = \frac{1}{m}\sum_{i=1}^{m}\frac{x(i) - VB(i)}{x(i)}. \tag{10}$$

This KPI also gives information about the amount of traffic which is unnecessarily sent from the sender. It is also used to calculate the cost of the adaptation.

3.3 System KPIs

In case of System KPIs the metrics have to be advanced by another dimension. j indicates each session of the network. It is also imaginable that different sessions are controlled by different algorithms. A session describes the end-to-end connection between two user equipments in the network.

3.3.1 Radio Subband Utilization

In mobile networks one target is to utilize the available spectrum or radio resources completely. The maximum available link bitrate mostly depends on the location of the users. If all users are in rather bad conditions, e.g. shadowing or long distance paths, the system will have a low cell throughput although the utilization of the network is high. To bring this into a KPI the subband utilization can be used, which is the percentage of used parts from the complete frequency band. The number of used subbands per user varies for LTE. So, a user can allocate more than one subband. If all subbands are used the system is completely utilized.

3.3.2 Spectrum Efficiency

The spectrum efficiency refers to the throughput which is transmitted over the given bandwidth. Bad channel conditions are considered in difference to the radio subband utilization. The KPI gives a statement, how efficient data can be transmitted. The algorithms are all evaluated in the same scenario with a fixed stochastic model, which is further discussed in Section 5.2.3. For the evaluation of spectrum efficiency the stochastic of the simulator was frozen to get an undisturbed statement. Since all algorithms suffer them in the same amount of bad channel condition, they do not play a role in the comparison between the algorithms. The spectrum efficiency is normalized by the number of active transmitters N_{Tx}. For the interval [1,m] it is calculated by [16]:

$$\eta = \frac{1}{N_{Tx} B} \sum_{j=1}^{N} \bar{x}_j \tag{11}$$

where B represents the available radio spectrum Bandwidth and $\bar{x}_j$ the average throughput for each session j, see equation (4). This indicator is normally taken to measure the efficiency of quantization in mobile radio technologies but in our case it is efficient to calculate the amount of data which can be transported with the control of an adaptation algorithm over the available bandwidth.

3.3.3 System Stability

The measuring of per-session metrics is only one aspect of algorithm stability. To see how an algorithm behaves a realistic scenario has to be set up. In this scenario multiple sessions which are controlled by this algorithm and also by cross traffic, which is not controlled by this

algorithm, have to be considered. The traffic should have varying rate over time on forward and reverse channels. In [17] a measurement of CoV for system stability is introduced. Two kinds of measurements were introduced; firstly a measurement of the throughput per session discussed in Section 3.2.1, secondly, a measurement of the average throughput of all sessions, sharing the same bottleneck discussed in this Section. If the system is not congested the second measurement will give information about the stability or instability of the whole system [17]. The coefficient of variance is defined as:

$$CoV_{system} = \frac{\sigma_{total}}{\bar{x}_{total}} \tag{12}$$

with the average system throughput:

$$\bar{x}_{total} = \frac{1}{N}\sum_{j=1}^{N} \bar{x}_j, \tag{13}$$

where $\bar{x}_j$ the average throughput for each session j, see equation (4). The standard deviation of the average system throughput in the interval [1,m] is

$$\sigma_{total} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} (\bar{x}_{total}(i) - \bar{x}_{total})^2}, \tag{14}$$

with $\bar{x}_{total}(i)$ indicates the throughput of all session at the step i; see following equation, where n is the number of all session in the system:

$$\bar{x}_{total}(i) = \frac{1}{N}\sum_{j=1}^{N} x_j(i) \tag{15}$$

[17] mentions that, if a system is congested, the stability of control protocols will improve. It is proven that this is applicable for the algorithms evaluated in this thesis (see Section 6.6).

3.3.4 Fairness

The overall statement of fairness is how the resources are divided under different sessions and especially under different algorithms. So fairness can be considered between sessions with the same algorithm and between sessions with different algorithms. For example, in [18] it's said that any algorithm which allocates zero throughput to any user is unfair, otherwise it is fair. This statement is rather categorical. For an evaluation of different algorithm a continuous indicator is needed.

Fairness Index

Let us consider

$$\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_j(i) \tag{16}$$

as the average throughput of session j over the interval [1,m]. Then Jain's fairness index for this interval is defined as [19]

$$F = \frac{\left(\sum_{j=1}^{N} \bar{x}_j\right)^2}{N\sum_{j=1}^{N} \bar{x}_j{}^2} \tag{17}$$

with N indicating all sessions. As shown in [19] this KPI is dimensionless, metric independent, bounded between 0 and 1, and it is continuous so that any slight change in $\bar{x}_j$ changes the index. An algorithm is maximally fare if $F = 1$, which means that all users allocate the same throughput. This index can also be used to measure other metrics as delay or packet losses.

The disadvantage of this index is that the general assumption that all sessions require the same amount of resource does not fit in the case of mobile networks. Channels which are in some way declined have to take more resources of the network to achieve the same amount of throughput or to deliver equal low delay.

Ɛ-Fairness

In [20] the max-min fairness is described: An allocation of throughput is max-min unfair if an increase of this throughput causes a decrease of any other session's throughput, otherwise its fair. This criterion again only allows a categorized statement (fair/unfair). To form a KPI out of this the min-max ratio $MINMAX$ of [19] can be used. This is the Ɛ-Fairness:

$$MINMAX = \frac{\min_i[x_i]}{\max_j[x_j]} = \min_{i,j}\left[\frac{x_i}{x_j}\right] = 1 - \varepsilon \tag{18}$$

Note that x_i and x_j are measured at the same time but are the throughput of different sessions. An algorithm is maximum fair if $MINMAX = 1$ and $\varepsilon = 0$. It measures the worst case of the system. This is consistent to the statement that any algorithm which produces a zero throughput is unfair [22]. The term will be maximized if all sessions, which share the gateway of the same bottleneck, receive the same throughput. The amount of resource allocation is not considered for the KPI so that it has the same disadvantage like the fairness index. However it is good to use it for evaluation, because the sessions which suffer the most are more intensively considered.

TCP Fairness

TCP-friendliness or TCP fairness is referring to fairness between a TCP session and a session controlled by any other algorithm [5]. The sending rate per second of a TCP session can be calculated as:

$$x = \frac{1.2}{RTT * \sqrt{L}} \tag{19}$$

with L the packet loss rate and RTT the round-trip time of the given session [21]. If the rate of the new algorithm is higher than the TCP rate, the algorithm will be TCP unfair. The fairness can be shown by comparing the throughput of a TCP session using the same bottleneck like a session with another algorithm.

Fairly Shared Spectrum Efficiency

In [16] the fairly shared spectrum efficiency (FSSE) is described as “a normalized measure of the minimum throughput that a terminal achieves”. In other words this means that the portion of spectrum efficiency, which is received by the receiver with the minimal average throughput, is estimated for all receivers. If the spectrum efficiency is quite high, because there are view sessions which acquire a huge amount of throughput, the FSSE is low if there are other sessions suffering. $FSSE$ is mathematically described in the interval [1,m] according to [16]

$$FSSE = \frac{1}{N_{Tx}B} \sum_{i=1}^{m} N \min_j \left[x_j(i)\right] \tag{20}$$

with the number of active transmitter N_{Tx}, B the available radio spectrum bandwidth and N the number of active sessions. If $FSSE$ is divided by the spectrum efficiency η the solution is:

$$\frac{FSSE}{\eta} = \frac{\sum_{i=1}^{m} N \min_j \left[x_j(i)\right]}{\sum_{j=1}^{N} \bar{x}_j}. \tag{21}$$

So, if all sessions gain the same amount of throughput $FSSE$ is equal to η. If the spectrum efficiency is maximized and $FSSE = \eta$, the system would be completely fair, because all users would receive the same bitrate and would completely utilize the network. $\frac{FSSE}{\eta_{max}}$ describes the system fairness with regard to the system utilization.

Chapter 4 Evaluated Algorithms

This chapter explains the three media rate adaptation algorithms. It also explains the fourth algorithm which is the proposal of this work for a media rate adaptation algorithm, called E-Sprout.

Media rate control techniques can be roughly classified into two main categories [22]:

- Rate and window based adaptation techniques where the rate is controlled by increasing or decreasing the rate in response to congestion signals like delay or loss.
- Available bandwidth estimation (ABE) techniques where the rate is directly estimated or forecasted as a function of congestion signals.

The investigated algorithms are selected because of their publicity and their recently development. The “Google Congestion Control Algorithm for Real-time Communication” (GCC) is published to RMCAT [4]. It is a mixture of the rate based congestion control and ABE. Sprout which is published by the Massachusetts Institute of Technology (MIT) is a purely ABE [2]. Self-Clocked rate adaptation which is an Ericsson proposal for conversational video rate adaptation technique is a window based approach. These three algorithms cope a big area of adaptation frameworks and significant conclusion can be given from their investigation.

In the case of Sprout, the algorithm didn’t fit into the conversational video scenario and is evolved to a version called E-Sprout. E-Sprout is the proposal of this thesis for an ABE which has a fast reaction on channel variation and uses the fast adaptation techniques from Sprout.

4.1 Framework Structure

Figure 1 shows the media rate adaptation framework. It is implemented in the clients and splits up in a sender and a receiver part.

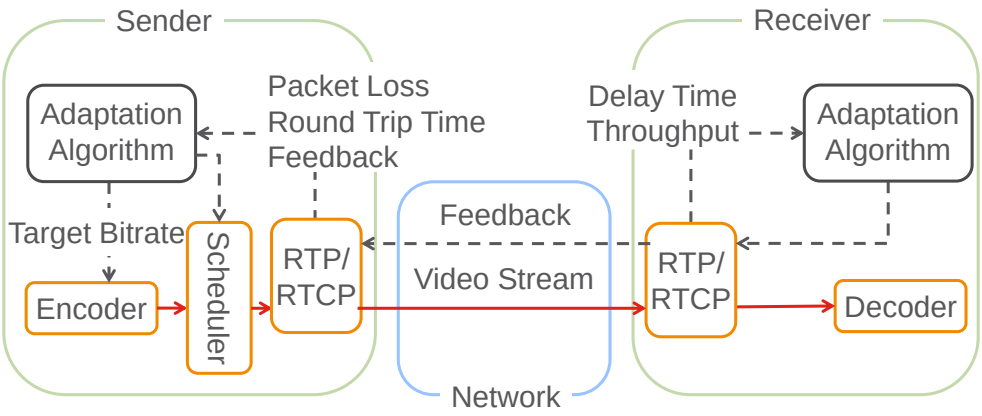

Figure 1 Adaptation framework showing sender and receiver components.

Different channel metrics are measured to detect congestion, e.g., throughput, RTT, packet loss rate and the time between two arriving video frames. The receiver estimates the channel and sends back a feedback to the sender. The sender is the controlling part in the framework and sets the target bitrate for the encoder.

In case of Sprout and E-Sprout, packets are queued in the scheduler block in order to protect the channel from congestion. The transmission of the video is done by the Real-Time Transport Protocol (RTP). For GCC the feedback is send over Real-Time Transport Control Protocol (RTCP) reports. For Sprout, E-Sprout and Self-Clocked the feedback is piggybacked on the media packets.

4.2 Goolge Congestion Control (GCC) Algorithm for Real-Time Communication

Google proposed the Google Congestion Control (GCC) in IETF (RMCAT WG) for real time conversational media. GCC is part of their WebRTC implementation code [4]. This algorithm is not explicitly developed for mobile radio application, but, through its publicity, will be part of many upcoming implementations.

GCC consists of a sender side controller and a receiver side measurement block. Figure 2 illustrates the adaptation Framework specified for GCC. The different blocks are discussed Section 4.2.1 and 4.2.2.

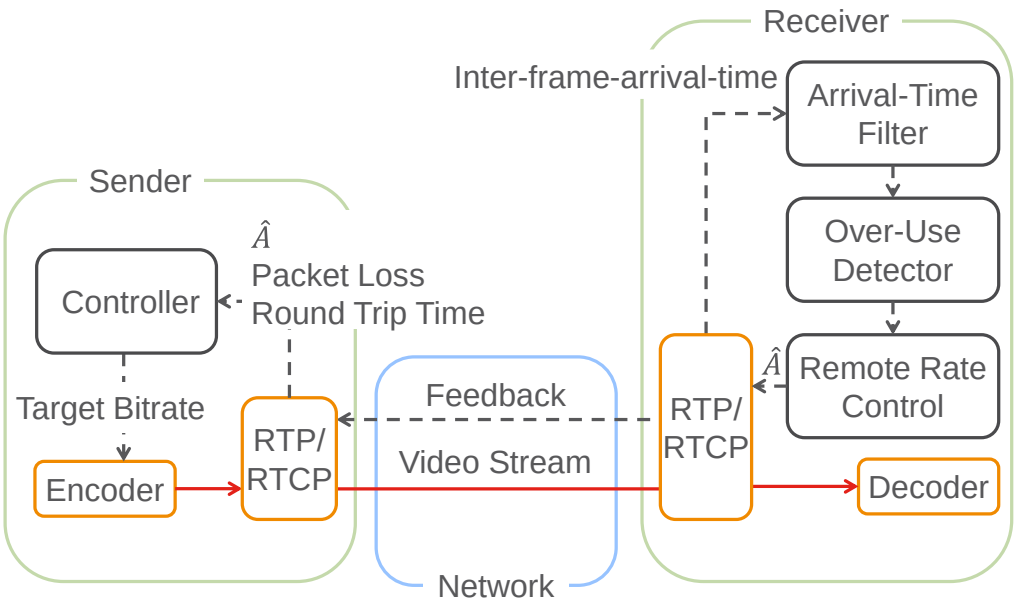

Figure 2 Adaptation Framework for GCC algorithm.

4.2.1 Receiver-side Control

The task of the function block, which is built into the receiver, is to estimate the maximal available bitrate of the channel. In order to fulfill its task, the function block consists of four main parts: an RTP entity, an arrival-time filter, an over-use detector, and a remote rate-control [4]. In the first instance, the RTP entity handles the arriving packets from the sender

and measures the arrival time. The transmitted frames own a timestamp. This timestamp is based on the system time of the sender and marks the time when the packets are transmitted. The timestamps and the arrival time are used for the arrival-time filter. The main purpose of this filter is to estimate the inner network conditions. Therefore, a Kalman filter is used [4].

Channel Model for GCC

For the estimation of the inner network conditions with the Kalman filter, a model of the channel is used. The model which is implemented in the GCC algorithm is defined as [4]:

$$d(i) = \frac{L(i) - L(i-1)}{C(i)} + w(i) = \frac{dL(i)}{C(i)} + w(i) = \frac{dL(i)}{C(i)} + m(i) + v(i), \tag{22}$$

where $d(i)$ is the inter-arrival-time between the frame i, and $i-1$. $d(i)$ is estimated by using equation (23), where $t(i)$ is the time of the arriving packet and $T(i)$ is the timestamp based on the sender side system time. Since only the differences of these times are used different bases in time are irrelevant. The arriving packets of one frame will all have the same timestamp. $L(i)$ is the size of the frame, $C(i)$ the capacity of the channel and $w(i)$ a stochastic variable which is normal distributed. $dL(i)$ is the difference between the size of two frames.

$$d(i) = t(i) - t(i-1) - \big(T(i) - T(i-1)\big) \tag{23}$$

In [4] it is announced that $w(i)$ is actually consisting of a mean $m(i)$ and a variation $v(i)$, like it is shown in equation (22). The mean $m(i)$ is meant to be growing if the number of packets between sender and receiver is growing in the network it is comparable to the self-inflicted delay by the algorithm. $v(i)$ is zero mean Gaussian process noise with variance $Var(v) = \sigma^2$.

Arrival-Time Filter

The main purpose of the arrival-time filter of GCC is to estimate the capacity $\hat{C}(i)$ and the mean $\hat{m}(i)$. In Figure 3 the Kalman filter [23] with the setup of GCC [4] is shown.

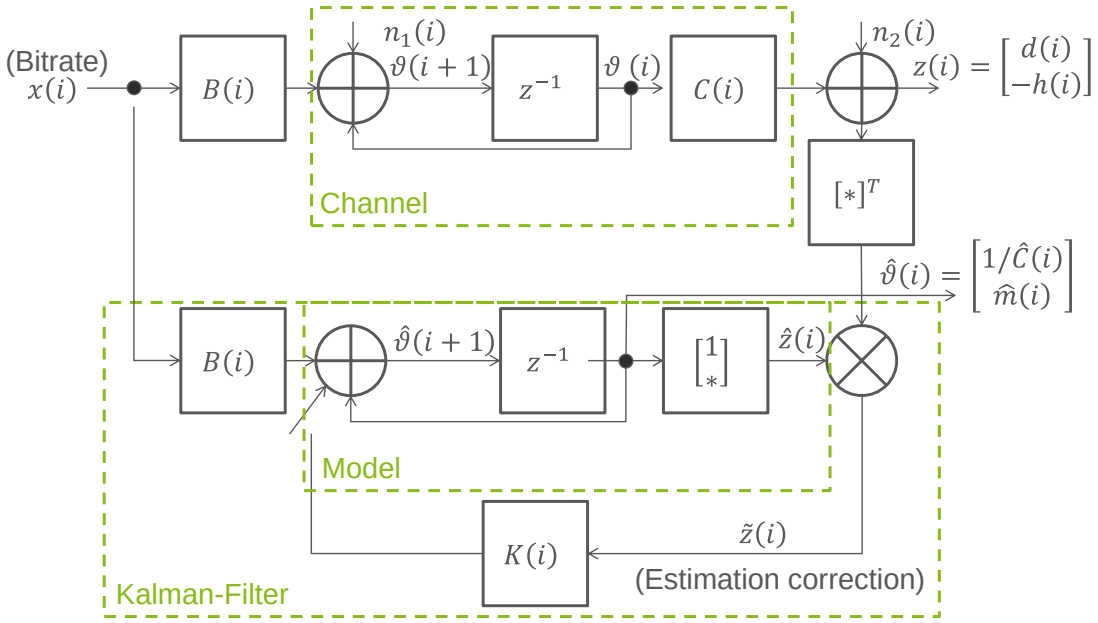

Figure 3 Kalman filter with GCC setup.

Depending on the bitrate $x(i)$ the frames have a specific inter-arrival-time $d(i)$ and a specific length $L(i)$ which can be measured with any frame arriving at the receiver. With the model of the channel the Kalman filter calculates the mean $\hat{m}$ of the variation and the capacity $\hat{C}$, while being corrected through $\tilde{z}(i)$. i indicates each frame. $n_1(i)$ is the variation $v(i)$ of the network condition, $n_2(i) = 0$. $h(i) = [dL \quad 1]$ where dL is the difference between the length of frame at step i and frame at step $i-1$. The Kalman Gain is defined as [4]

$$K(i) = \frac{E(i-1) * h(i)^T}{\overline{Var(v)} + h(i) * E(i-1) * h(i)^T} \tag{24}$$

with

$$E(k) = (\mathrm{I} - \mathrm{K}(i) * h(i)^T) * E(i-1) + Q(i) \tag{25}$$

where $Q(i)$ is the covariance of $x(i)$ [4].

Over-use Detector

The estimated mean $\hat{m}(i)$ is used by the Over-use detector. In Table 1 the different states and their underlying reasons are shown. $\boldsymbol{t_{m>\gamma_1}}$ is the time since the mean $\hat{m}(i)$ is above $\boldsymbol{\gamma_1}$. $\#Frames_{\boldsymbol{m>\gamma_2}}$ is the number of frames since the mean $\hat{m}(i)$ is above $\boldsymbol{\gamma_1}$. The thresholds used for this thesis are $\boldsymbol{\gamma_1} = 2.5, \boldsymbol{\gamma_2} = 100ms, \boldsymbol{\gamma_3} = 1$.

Comparison	State
$(\hat{m}(i) > \gamma_1)$ && $(t_{m>\gamma_1} > \gamma_2)$ && $(\#Frames_{m>\gamma_1} > \gamma_3)$	Over-use
$(\hat{m}(i) < \gamma_1)$ && $(t_{m<-\gamma_1} > -\gamma_2)$ && $(\#Frames_{m<-\gamma_1} > \gamma_3)$	Under-use
!Over-use && !Under-use	Normal

Table 1 States for Over-use detector.

In words, if $\hat{m}(i)$ is above the threshold $\gamma_1$for a time $\gamma_2$and a number of frames γ_3, an over-use will be triggered. If $\hat{m}(i)$ is under the threshold $-\gamma_1$ for a time γ_2 and a number of frames γ_3 an under-use will be triggered. If neither an over-use nor an under-use is triggered the detector will signal the normal state [4].

Remote Rate Control

The purpose of the rate control is to estimate the available bitrate $\hat{A}$. Therefore, the state of the over-use detector is used. Figure 4 shows the state chart of the rate control [24].

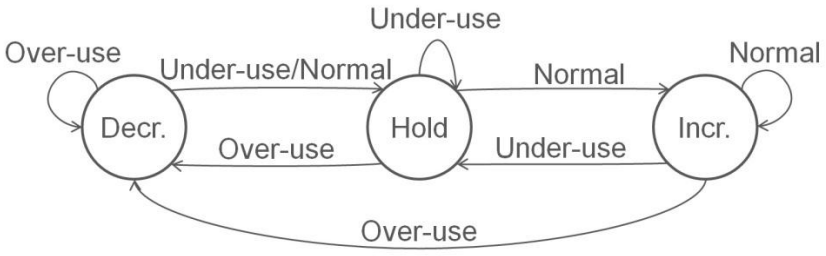

Figure 4 State chart of rate control with signals from over-use detector [24].

As long as the state of the over-use detector is normal the rate control will increase $\hat{A}$. This makes the algorithm more aggressive and assures that the rate control will reaches the available bitrate A of the channel [4].

$$\hat{A}(i-1) = \eta * \hat{A}(i-1) \tag{26}$$

If an increase is triggered $\hat{A}$ will increase with the factor η which is a function of the RTT and $Var(\hat{v})$ like it is shown in equation (26). Equation (27) shows the factor. B, b, d, c_1, c_2 are design parameters [4].

$$\eta(RTT, Var(\hat{v})) = \frac{1.001 + B}{1 + e^{\left(b\left(d*RTT-(c_1*Var(\hat{v})+c_2)\right)\right)}} \tag{27}$$

$\hat{A}$ has to be bounded. To do so the GCC algorithm calculates the incoming bit rate $\hat{R}(i)$ over a T seconds window like it's shown in:

$$\hat{R}(i) = \frac{1}{T} sum\big(L(j)\big) \qquad \text{for } j \text{ from } i - N_i \text{ to } i \tag{28}$$

Here $L(j)$ is the payload size of the frame j, and $N(i)$ is the number of frames received in the past T seconds. A recommendation of [4] is to choose this window size in the range of 0.5 to 1 second. In the implementation evaluated in this thesis the window size is 1 second. It is matched to the rate control at the sender side. The upper bound of the estimated bitrate $\hat{A}$ defined as:

$$\hat{A}(i) < 1.5 * \hat{R}(i) \tag{29}$$

In case of an over-use and a transition to the decrease state GCC will decrease $\hat{A}$ with a fixed factor α like it is shown in:

$$\hat{A}(i) = \alpha * \hat{A}(i-1) \tag{30}$$

The factor α is between 0.8 or 0.95. It is 0.95 if $\hat{R}$ is above the known maximal estimated incoming bitrate $\hat{R}_{max}$ or if the rate control is in the decrease-state, otherwise the factor is 0.8. The rate $\hat{R}_{max}$ is only measured if the rate control is in the hold-state and is 0 if the rate control is in increase-state. If an under-use is triggered to the rate control subsystem, the queues in the network are running empty. The available bitrate estimation $\hat{A}(i)$ is lower than the actual available bandwidth $A(i)$. In order to empty the queues in the network the rate control subsystem will enter the hold state, and will stay there until the normal state has been triggered from the over-use detector. This guarantees that the queue are cleared and the delay is as low as possible. A good guess of the available bitrate, if the normal state is entered, is $\hat{R}_{max}$, which is measured through

$$\hat{R}_{max} = \max_{i}\left[\hat{R}(i)\right] \tag{31}$$

where i is indicating the steps between the time the subsystem entered the hold state and the normal state. The system returns directly to the bitrate, at which the under-use was firstly triggered. The rate control now sends $\hat{A}(i)$ to the sender through RTCP. This messages are send in a “heartbeat”. In the other opportunity to send only if a change occurs, the packet could be lost and the system will react too slow. Also the loss of such a heartbeat triggers a reducing of the send rate. If a significant change is estimated the new bitrate will be send immediately without waiting, however the frequency of this fast information is bounded to an upper level. This upper limit is calculated through $t_{min-fb-interval}$ and the maximum interval level is set to $t_{max-fb-interval}$, which are set by the RTCP bandwidth.

4.2.2 Sender-side Control

The estimated available bitrate $\hat{A}(i)$ information is send from the receiver to the sender and is used from the congestion controller. Together with the round-trip time and the packet loss the available bitrate $\hat{A}_s$ is estimated, which is the target bitrate for the encoder. This control is performed every time a receiver report arrives. If no report is received for longer than $2 * t_{max-fb-interval}$ the algorithm will take action as if all send packets in this interval where lost. The result is a halving of $\hat{A}_s$. On sender side the packet loss information gives the first clue. If the packet loss rate since the previous report is between $2-10\%$ the estimated available bitrate $\hat{A}_s$ is kept the same. If it is higher than 10% it will be lowered through:

$$\hat{A}_s(i) = \hat{A}_s(i-1) * \left(1 - \frac{L}{2}\right) \tag{32}$$

with L is the packet loss ratio. If the packet loss rate is lower than 2% since the last report $\hat{A}_s$ is increased by

$$\hat{A}_s(i) = 1.05 * \left(\hat{A}_s(i-1) + 1000\right). \tag{33}$$

In the next step $\hat{A}_s$ is bounded. The floor is the TCP Friendly Rate Control (TFRC) formula (see equation (34)) and the ceiling is the receiver-side estimated available bitrate $\hat{A}(i)$.

$$\hat{A}_s(i) >= X_{Bps} = \frac{8 * s}{RTT\sqrt{2 * b * \frac{p}{3}} + t_{RTO} * 3\sqrt{3b * \frac{p}{8}} * p * (1 + 32p^2)} \quad (34)$$

X_{Bps} is the average sending rate in bits per second [25]. RTT is the round-trip time in seconds. b is the number of packets acknowledged by a single TCP acknowledgement (set to 1 per TFRC recommendations), t_{RTO} is the TCP retransmission timeout value in seconds (set to $4 * RTT$) and s is the average packet size in bits per second.

4.3 Self-Clocked Algorithm

The Self-Clocked algorithm evaluated in this thesis was built in advanced by Ericsson directly in the framework of the system simulator. The motivation for this algorithm was to lower the latency over best effort LTE bearers even for high video bitrates. The scheme utilizes functionalities from TCP LEDBAT [26], TFWC [27], SST [28] and Paceline [29]. The principle of this algorithm is that every packet which is arriving at the receiver is acknowledged to the sender. This acknowledgement is send back by piggybacking it onto its own frames. In Figure 5 the algorithm components are drafted. Here the frame scheduler controls the number of frames send to the network queues. The Network congestion control serves to set an upper limit on how much data can be in the network. This data is named as bytes in flight. The upper limit is calculated by the one way delay (OWD) of the channel. It is named as congestion window $(CWND)$. This $CWND$ is further described in 4.3.1.

The rate control gives a direct feedback to the video encoder and controls the target bitrate. The measurement variable for this control is the age of the frames which are stored in the queue before the frame scheduler. The reaction to a sudden overflow of this queue has to be prompt because this queue is also measured in the end-to-end user delay.

The receiver side block of the algorithm consists of a block which piggybacks the acknowledgement on to the packets which are transmitted from receiver to sender.

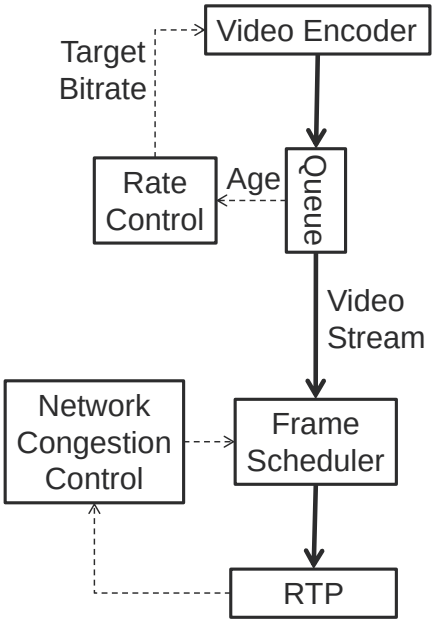

Figure 5 Self-Clocked algorithm components overview sender side.

4.3.1 Packet Conservation

Self-Clocked algorithm is using packet conservation to prevent the inner network queues to overflow and drop packets. In order to do so the frame scheduler follows the $CWND$ target. The algorithm uses the OWD to calculate the $CWND$. This is done similar to the LEDBAT concept. The $CWND$ will increase if the measured OWD is under a predefined target, otherwise it will decrease. This delay target is typically set to 50-100ms.

The $CWND$ is used by the frame scheduler to calculate the number of bytes to send into the network. The decision if a frame is send is done by:

$$If(lastTransmittedBytes - lastAckedByte < CWND)$$
$$Transmit\ next\ frame.$$

Here $lastTransmittedBytes$ is the number of bytes already sent to the receiver and $lastAckedByte$ is the number of bytes received at the receiver when the last acknowledgement response, which has arrived at the sender, was produced. If an acknowledgement arrives at the sender the $CWND$ and the $lastAckedBytes$ are always updated. So an acknowledgement consists of a list of sequence numbers of the frames which were received since the last produced acknowledge. This information consists of 8 bytes. The $CWND$ is allowed to increase even though it is not fully utilized. This can happen if the encoder bitrate is lower than maximum bitrate A of the channel. The upper limit of $CWND$ is the number of bytes in flight which always have to be greater as a fraction of the $CWND$. This ensures that the $CWND$ is always proportional to the actual sent bitrate, but is not too small if the bitrate increases from one time to another. The $CWND$ will also decrease if the bitrate of the encoder decreases because there is simply nothing to send. This can often happen with variable bit rate videos, where temporal bitrate can vary very much.

4.3.2 Switch Sensitiveness of Loss or Delay Detection

Another important feature of the algorithm is that it can switch between delay sensitive and loss sensitive. If a packet drop occurs the delay target is increased so that the OWD is no longer the controlling feature. This is done if packet drops are accrued due to other competing large file downloads which typically use TCP algorithms that are only sensitive to loss. In this case, the TCP session would always eliminate the self-clocked session. The switching is done if a packet loss occurs and it will switch back if no packet loss is detected for a prolonged period.

4.3.3 Packet Pacing

Packet pacing is also included into the Self-Clocked algorithm. Packet pacing tries to minimize coalescing of packets, i.e. if the $CWND$ is quite high and the queues of the network are empty the algorithm will send a burst of packets. In order to ensure that each packet has enough time to be transmitted from the radio interface and to avoid packet losses and increased jitter a time interval is enforced between each packet transmission. The time interval is calculated from the $CWND$ and the estimated RTT.

4.3.4 Exceed Congestion Window If OWD is Low

If the OWD is low the network queue seems to be empty. The probability that a high amount of data will overfill the channel is lower than in the case with high OWD. So for a short time of a frame the bytes in flight limited through the $CWND$ can exceed this limit. In the case of VBR video or for an I-frame transmission this feature is meant to improve the performance. On the other hand if the network is congested it can also cause delay spikes.

4.3.5 Exponential Start Behavior

Self-Clocked has a startup state to have a small startup. The time to reach the available bitrate, if the channel is not congested, when the session starts is kept low with this state. In this state the growing of the target bitrate, which is normally fixed through a factor, is done through an exponential grow. This guarantees, that the user doesn’t wait too long until the algorithm lasts in its steady state.

4.4 The Sprout Algorithm

The implementation of Sprout, which is evaluated in this thesis, is designed to compromise two objectives. While striving for the highest possible throughput, packets shall be prevented from waiting too long in a network queue [2]. It was invented from the MIT and presented at [2].

The implementation had to be translated into the system simulator language. Also the connection to the environment is done in another way in the simulator and had to be changed. The bulk transfer scenario which is further discussed in Section 5.2.4 was taken to evaluate the implementation and it showed the same behavior as the real Sprout implementation which can be seen in Section 6.10.

In [30] it is mentioned that in cellular networks the RTT varies with transient latency spikes to hundreds of milliseconds, throughput varies by a factor of 10 in a short time scale, and buffers do not drop packets until they contain 5-10 seconds of data at bottleneck link speed. Because of that Sprout is not designed to take packet losses as a measurement of the available bitrate of the channel, and the fullness of in-network queues. It only measures the received throughput and forecast this to a 160 ms window.

Figure 6 illustrates the structure of the framework in which Sprout is evaluated. The video data is queued up before it is transmitted to the network. The rate control takes *BytesToSend* which is the result from the adaptation algorithm and is discussed in Section 4.4.3, to set the target bitrate for the encoder.

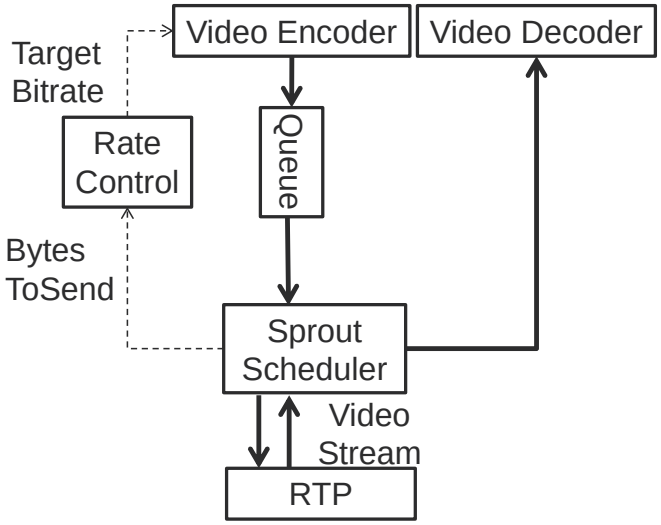

Figure 6 Sprout framework overview.

4.4.1 Channel Model

Sprout's model of the channel is a stochastic Poisson process. In the time of creation a long measurement with a stationary cell phone was taken and a curve was fit to the probability of inter-arrival time of the packets at the receiver. This distribution describes a Poisson process in approximation. The underlying rate λ can be estimated by measuring the throughput on receiver site. The stochastic process can then be used to forecast the number of bytes which should be transmitted from the sender.

The rate λ is changing frequently over time, so the receiver has to transmit an update. This update cannot take place immediately, because it has to be sent through the network path as well. Hence, the variation of λ is forecasted firstly. A Browian motion is taken to forecast λ. This makes the whole model a double stochastic process in which λ is a stochastic process itself. The Browain motion has an underlying noise power of σ. This means that the forecasted λ gets more unreliable in every step, in which the sender doesn't get an update from the receiver [2]. Figure 7 illustrates this model.

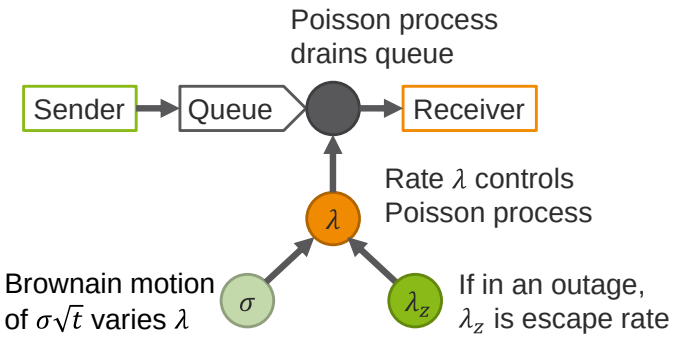

Figure 7 Sprout's channel model [2].

If Sprout's forecast of the channel indicates that no data shall be sent, Sprout's receiver will estimate $\lambda = 0$. Consequently there would be an outage for the next steps, because no packets would arrive at the receiver anymore. In order to reinitialize the traffic, an exponential distribution $e^{-\lambda_z}$ is used to calculate the duration of this outage. This serves to match the behavior of real links, which do have "sticky" outages in the experience of [2].

4.4.2 The Forecast

For all feasible λ, which are in a special range, their probability distribution functions are calculated in advanced.

Sprouts step size to forecast the throughput is called tick. Every tick in the implementation which is evaluated in this thesis is 20 ms long. In each tick the receiver counts the number of packets completely received in this tick and multiply the list of feasible λ with the probability that this λ was the source of the amount of packets. Let's assume that τ is the tick length and k are the number of bytes observed during the recent tick. Then

$$F(\hat{\lambda}) = P_{old}(\hat{\lambda}) * \frac{(\hat{\lambda} * \tau)^k}{k!} e^{-\hat{\lambda}*\tau} \tag{35}$$

is the new non-normalized probability that the current rate measured $\hat{\lambda}$ describes the real channel λ. The estimated probability has to be normalized so that the addition of all probabilities of the feasible λ sums up to unity. This is called Bayesian updating [2]. This probability is later added to the probability distribution function of each rate λ to do the forecast. This probability distribution function describes the probability of the number of MTU sized packets for a rate λ.

The receiver forecasts the number packets which will arrive in the next 160 ms or in the next 8 ticks. In order to do so, the number of packets arriving in each tick is calculated. For every number of packets, called "count" from Sprout, the probability of this count is weighted with the probability of this rate λ. Table 2 illustrates the weighted count probabilities. Then the weighted count probabilities are summed up over all rates λ for each count, as shown in

$$count\_prob(i) = \sum_{\lambda=0}^{N} a_\lambda * pdf_\lambda(i) \tag{36}$$

with a_λ the probability of each λ calculated through Bayesian motion and $pdf_\lambda(i)$ which is the probability of count i of the Poisson distribution function with rate λ. The count which has the minimum sum of weighted count probabilities and is greater than 5% is the requested one and can be taken as number of MTU sized packet received for this tick.

Rate\Count	1	2	3
$\lambda = 0$	$a_0 * pdf_{\lambda=0}(1)$	$a_0 * pdf_{\lambda=0}(2)$	…
$\lambda = 1$	$a_1 * pdf_{\lambda=1}(1)$	…	
$\lambda = 2$	$a_2 * pdf_{\lambda=2}(1)$		
$\lambda = 3$	$a_3 * pdf_{\lambda=3}(1)$		
⋮	⋮		

Table 2 Weighted Count Probabilities.

The difference to the next higher tick is that it is more probable that more packets had arrived till this tick.

The number of counts per tick can for example look like the first row in this table the other rows will be discussed in 4.4.3:

	Tick 0	Tick 1	Tick 2	Tick 3	Tick 4	Tick 5	Tick 6	Tick 7
Count	1	3	6	8	10	13	16	18
QueueSize	12	13	12	10	8	5	2	0
To Send	13	4	2	0	0	0	0	0

Table 3 Example of count received per tick, queue size of network and count send per Tick.

This forecast of the number of packets received in the future is named *packet delivery forecast.* In the next step the receiver sends this forecast back to the sender by piggybacking it onto its own outgoing packet. In addition this packet also contains the number of already received packets, so that the sender knows how many packets are still in the queue.

4.4.3 Using the Forecast

If the sender receives the forecast it calculates the number of packets save to send in the next tick. In order to do that it calculates:

$$cumulative\_delivery\_forecast(i) = count(i+5) - count(i) \tag{37}$$

In words this means that the number of packets which is allowed to be in the network queue in recent tick i is calculated with the forecast of the received packets for the tick i subtracting it from the forecast of tick $i + 5$. The result for every tick is shown in the second row from Table 3. The number 5 is taken because Sprout wants to forecast the number of packets save to send over the channel, so that they will arrive in the next 5 ticks or 100 ms.

In order to calculate the packets which can be send in the recent tick, Sprout calculates the number of packets which are in the queue at the recent tick an subtract this number from $cumulative\_delivery\_forecast(i)$. This is done through

$$queue(i) = (count_{rec} + count_{estimated-rec}(i)) - count_{send}(i) \quad (38)$$

where $count_{rec}$ gives the total number of packets, which were received at the receiver before the last forecast, which is received at the sender, was produced. $count_{estimated-rec}(i)$ gives the estimated number of packets received at the receiver since the last forecast. $count_{send}(i)$ gives the total number of already send packets. The result of $\big(cumulative\_delivery\_forecast(i) - queue(i)\big) * MTUsize$ is named $BytesToSend(i)$ and it is shown in row 3 of Table 3 for the example forecast. $MTUsize$ is the maximal transfer unit size and is 1500 bytes in this thesis. Figure 8 illustrates the example forecast. The red line labels the packets which already arrived at the receiver and the black arrows illustrate $BytesToSend$. For simplicity $BytesToSend$ was transferred into number of packets by dividing it through $MTUsize$.

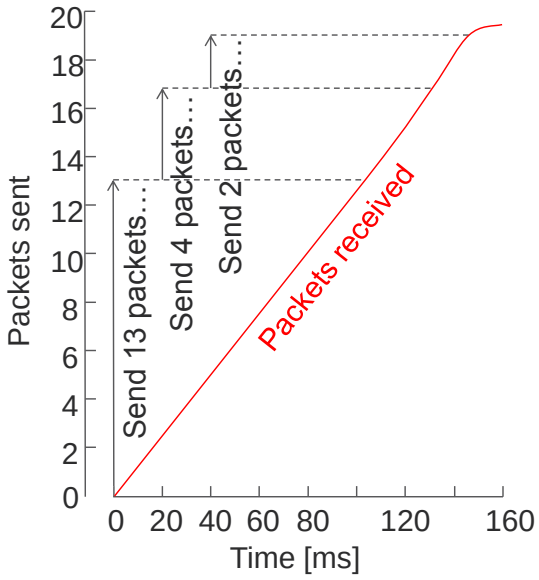

Figure 8 Illustrating example forecast [2].

Through this technique every packet which is send should have the probability of 95% to clear the channel queue within 100 ms. If a new forecast arrives Sprout set back the tick count to 0 and calculates $cumulative\_delivery\_forecast$ and $BytesToSend$ with the new forecast.

4.5 E-Sprout

The presented features in this chapter are built above the original code from Sprout, so E-Sprout is an evolved version of Sprout which improves Sprout for conversational real-time video traffic. It was developed after the evaluation of the other algorithms. Some ideas from Self-Clocked where taken to evolve E-Sprout as well. Sprout claimed to achieve the highest video bitrate while keeping the OWD at a desirable level. As shown in Section 6.10 this is not valid for conversational video. However, the performance of Sprout was insufficient, the

idea of the double stochastic process and throughput as a measurement is still promising to lead to good results. That is the reason why it was decided to improve Sprout. Figure 9 illustrates the data flow in E-Sprout. The red marked objects show the difference to the original version of Sprout. They will be discussed in the next chapters.

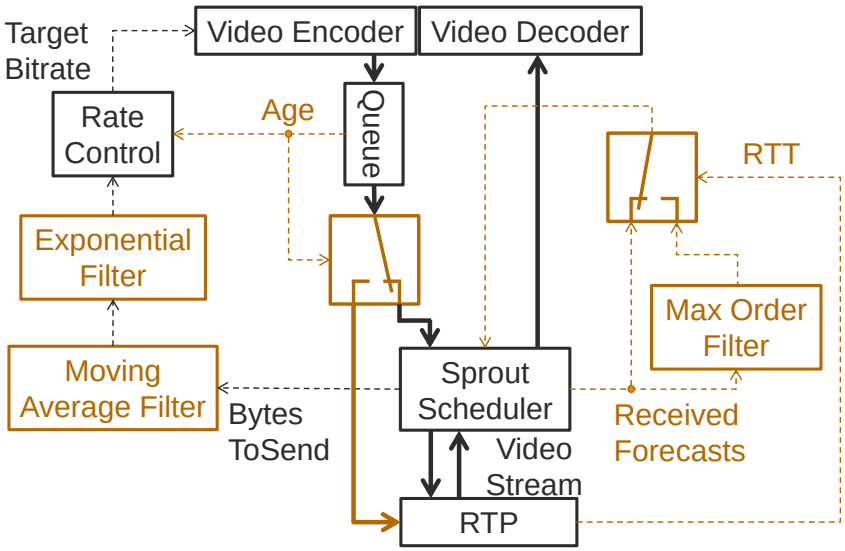

Figure 9 E-Sprout data flow.

4.5.1 Filtering of *BytesToSend*

In E-Sprout the rate control uses *BytesToSend* to estimate the target bitrate. Firstly, to be sensible to the numbers of bytes, which are waiting in the queue for transmission, E-Sprout subtracts them from *BytesToSend*. Then BytesToSend has to be filtered. For the rate control, *BytesToSend* is very unstable. E-Sprout filters *BytesToSend* through a moving average and then again through an exponential filter. The moving average filter subtracts peaks. The exponential filter provides smoothness when sudden changes of *BytesToSend* occur. Since the filtered *BytesToSend* is valid for one *tick*, E-Sprout advances it by normalizing and multiplying it with the frame period.

4.5.2 The Influence of the Age of Waiting Packets on *BytesToSend*

In addition, the rate control uses the age of the oldest packets in the sending queue to estimate the target bitrate. If packets wait longer than the frame period, *BytesToSend* will be decreased. This makes the algorithm more careful concerning packets which are waiting for transmission. The amount of decrease is influenced by the age of the packets. The higher the age is, the higher the amount of decrease is. If the age of the oldest packet is less than the frame period it is possible to queue up more data, so the target bitrate will be increased. This increase is proportional to the RTT. The result is the target bitrate, but it is still varying in small scale. To prevent the encoder from changing the bitrate too frequently, the minimum time between two changes is set equal to RTT.

4.5.3 Max-Order-Filter of the Arriving Forecasts when Low RTT

As shown in Section 6.10 a problem of Sprout is the false detection of congestion. To handle this, E-Sprout adds the RTT of the channel to the estimation. If the RTT is less than 100 ms, the channel is not congested. Thus, Sprout measures the video bitrate instead of the available channel bitrate. In this period of time E-Sprout uses a max order filter. It takes one of the last eight forecasts, which promises the highest bitrate. This ensures that the algorithm achieves the highest possible bitrate, if the channel is not congested.

4.5.4 Slow Start Behavior

At the beginning of a session, Sprout is quite optimistic and sends a high data rate, although it has not yet received a forecast and has no knowledge of the channel condition. To prevent congestion at this time, E-Sprout sends the minimal video rate until the second forecast of the receiver has arrived. Thus, E-Sprout achieves a slow start behavior. The forecast estimated at the beginning prognosticates a rather bad channel condition, because the sender only sends the minimum bitrate and the receiver measures the video bitrate instead of the available channel bitrate. After the second forecast has arrived E-Sprout sends more than the minimal bitrate and the next forecast prognosticates a better channel condition. So the amount of data, which is sent will grow slowly.

4.5.5 Added Frame Period to Fallback Time

In Sprout every sent packet has a fallback time, marking the time when the next packet should arrive at the receiver. It is known that there will be no data to send between video frames, so the time, until the next frame will be produced by the encoder, is added to the fallback time. The receiver will ignore if no packet arrives until the fallback time has run out.

4.5.6 The Sending of Packets during Congestion

The age of the packets, which are waiting for delivery, can be seen as pre-delay before transmission. This time is directly added to the OWD and downgrades the user experience. To prevent a high pre-delay the packets are not queued up longer than two times of frame period. Afterwards, they are directly sent, although E-Sprout is estimating bad channel congestion. This feature also softens false congestion detection. There is an upper limit how much data can be sent in such a time. If the congestion is lasting for a longer time, it will stop sending data after the amount of allowed packets is reached.

4.5.7 The Drop of too Old Packets

If the packets, which are waiting for transmission, are older than 220 ms the algorithm will drop the packet. This leads to higher packet loss. However, when there is congestion, the time a high OWD is present is heavily reduced. The result of this effect can be seen in Section 6.12 in Figure 30.

Chapter 5 System Design

This chapter lists the assumptions which were made for evaluation and defines the evaluation framework. The algorithms are assessed in heterogeneous environments containing both wired and wireless paths. The evaluation on the wireless paths takes place on a multi user radio system simulator with detailed models for the LTE physical layer, protocol layers, and radio resource management. The algorithms are evaluated for a range of channel characteristics such as different end-to-end capacity and latency and varying amount of cross traffic on a bottle-neck link like suggested in [3].

5.1 Reality and Assumptions

TCP Algorithms not Suitable

Congestion control is needed for all data transported across the Internet, in order to promote fair usage and prevent congestion collapse. TCP algorithms are not suitable for OWD and packet loss sensitive application [6]. Although TCP algorithms have different characteristics then the evaluated algorithms same KPIs can be taken for evaluation.

Traditional TCP congestion control requirements were developed for bulk transfer of non-time-critical data for the Internet. Examples are transfer of large files and smaller chunks of data in "as fast as possible" mode (e.g. Web pages). Also media “streaming” in a non-interactive manner is possible with traditional TCP algorithms, since the data is received when the user wants it but the exact timing of the delivery is not critical [6].

Real-time interactive media needs continuous data in a very limited time window. Buffers at the receiver side are very small, compared to buffers in the “streaming” non-interactive case. The absence of data in short time is directly noticeable by the user. Since sending “future”-data is not an option the amount of data which can be sent in short time is limited by the encoder. Another limiting factor is that lately delivered data is useless. To have a video call with a high OWD is rather unattractive for the user and it may be better to turn off the video completely [6]. No end-to-end protocol can provide low-OWD service when a network queue is already full of packets from another service [30], but the self-inflicted OWD is affected by the algorithm.

Queues of the Internet

If the timely transmission of packets is not possible they have to be queued or dropped. If the internet packets get queued up and if the buffers of the involved internet node run full the packets will be dropped. There is a controversy over how to act on the buffers of internet. Firstly, [4] argues that if queues are very short, over-use of the channel will only be visible through packet losses. Consequently, a media rate adaptation algorithm should react on packet loss. [30] points out that measurements indicate that buffers in cellular networks do not drop packets until they contain 5 to 10 seconds of data at bottleneck link speed. This means that a packet drop is only measured after the delay has grown unacceptably high. This opinion is also shared by [31], which claims that packet loss cannot be taken as an

indicator for congestion in cellular networks because large buffers produce bufferbloat when they do not have active queue management (AQM). For this thesis packet loss as a metric is not considered due to the last two arguments. Although Self-Clocked and GCC react on packet loss an algorithm has to work well with the absence of this metric. This assumption is also valid vice versa. So, it is suggested to run a simulation with limited small buffers or AQM in future work. AQM is a technique to drop packets before the queue is full. However it is not used in this evaluation, since it is only valid for special cellular networks. In this evaluation, packets are only dropped from the network when the amount of packets is so high that it contains 5 to 10 seconds of data at bottleneck link speed.

Bearer in Cellular Networks

The 3GPP Quality of Service (QoS) architecture for LTE and EPS (Evolved Packet System) uses bearers to handle different traffic types according to their QoS classes. Different bitrate and delay requirements are used for example for real-time audio and web browsing. Real-time conversational audio and video streams use dedicated bearers, for which a guaranteed bitrate, a maximum bitrate, delay and packet loss thresholds are defined [32]. Even within this framework and using dedicated bearers congestion for high bitrate video can occur. To keep an initially high video bitrate for a user will be very expensive and will consume a big part of the radio resources, if the link conditions for this user become worse. 3GPP defines the usage of explicit congestion notification (ECN). ECN requires managed network access. Therefore, ECN is not used in this evaluation.

Recently the usage of best effort bearers guaranteeing no fixed bitrate, OWD or packet loss thresholds for audio and video has gained popularity. Best effort bearers are used for RCS telephony, for Over-The-Top services, and for unmanaged access networks, e.g., Wi-Fi. A very low fixed video bitrate and quality would have to be used to guarantee timely transmission if best effort bearers were used. Bitrate adaptive video on the other hand allows using best effort bearers and the maximum possible video bitrate and quality. Best effort bearers are used in this work to evaluate the adaptation algorithms for video in LTE.

Experience of Video and Encoder behavior

The user experience of the video and thus the adaptation is evaluated through assumptions which were made. More important than the high of the video bitrate is the constancy of video quality. However the limit of OWD is the most critical part. If an algorithm delivers a slightly varying video with an OWD under the limit of 400 ms and a packet loss rate less than 2%, the assumption is that the user experience is good. Also packet losses rates which are higher than 2% but in total only appear for less than 3 seconds during a call which is 60 seconds long are negligible.

The encoder and decoder for the video are simulated by the system simulator. The time for encoding and decoding is fixed set to 40 ms. The encoder adapts directly and exactly if the adaptation will be in between the minimum and maximum video bitrate. Otherwise it directly delivers the minimum or maximum rate respectively. If the packets loss rate is over 2% the encoder has to send a new I-Frame to reinitialize the video stream.

5.2 Scenarios

The following table gives an overview which the KPIs is used for which evaluation issue and in which section the results of this evaluation can be found. The scenarios are further described in section 5.2.1 - 5.2.4.

Scenario	KPI used	Evaluated issue	Result section
LTE best effort bearer	OWD	Is the algorithm able to ensure the OWD limit?	6.1
	Frame/Packet loss rate and video bitrate	How good is the video quality?	6.1 and 6.14
	Radio subband utilization and spectrum efficiency	Does the algorithm use the available resources?	6.1 and 6.6
	Oscillation and stability	Is the instability or the conservativeness of the algorithm the reason for misbehavior?	6.5
	Overhead	How much control data has to be transmitted to ensure that the algorithm can work?	6.2
	Responsiveness	How fast does the algorithm react on congestion?	6.4
	System stability	Is the system stability influenced by the adaptation algorithm?	6.6
	TCP Fairness	Are TCP flows suppressed by the algorithm?	6.8
	Fairly shared spectrum efficiency	How much of the system resources are shared in a fair way between the users?	6.7
Monte Carlo simulation	OWD, video bitrate, packet loss rate and aggressiveness	Qualitative explanation of the behavior of the algorithm. • Start behavior • Reactivity to sudden congestion and to available bandwidth • Packet dropping • High Start Bitrate	6.3, 6.12, 6.13 and 6.16
	OWD and packet loss rate	Which are the operational ranges of the algorithm?	6.9
	Smoothness	How does the algorithm behave in a steady state scenario?	6.5

Scenario	KPI used	Evaluated issue	Result section
Bitpipe	Fairness index, Ɛ-Fairness	How strong is the influence of the algorithm on the fairness between the users	6.7
	Throughput	Which algorithm aggregates the highest bitrate in competition with the other algorithm on the same bottleneck	6.8
Bulk transfer	Inner Sprout algorithm indicators	Does the Sprout algorithm work with non-conversational data properties?	6.10

Table 4 Overview of scenarios and evaluation issues.

5.2.1 Bitpipe

The first scenario simulates a network with a fixed channel bitrate, which in this thesis is called “bitpipe“. In this scenario the mobile clients are fixed to the network, so the radio is not simulated. The scenario is used in different purposes. This scenario evaluates the ramp-up to the bottleneck capacity and the smoothness of the algorithm [3].

If the bitrate is set to infinity, it will be impossible that congestion occur. The algorithms sending rate is not limited. Since the video bitrate is still limited the throughput will not be infinitive because all evaluated algorithms try to minimize the overhead. This scenario is used to evaluate if the algorithm can cope with a totally uncongested situation.

If the bitrate is rather low, network congestion will occur frequently. This setup gives a conclusion about the fairness between all users. The amount of throughput for each user is only depending on the algorithm itself. It is also used to evaluate the fairness between different algorithms. In this setup all algorithms have the same amount of users.

5.2.2 Monte Carlo Simulations

The main motivation for this Monte Carlo simulation is to determine under which circumstances the algorithms will break down and also determine the operational range of the algorithm like it is suggested in [3].

The Monte Carlo method is used to evaluate the reaction of the algorithms to sudden available bitrate changes. Figure 10 sketches the method. The time until an algorithm is stabilized is set to 30 seconds. The start bitrate as well as the end bitrate are varying between 0.2 Mbps and 2 Mbps. It is possible that the end bitrate is higher than the start bitrate. This simulates a channel which is freed after congestion. The gradient of the slope changes because of the variations of the time of change. In the case of Monte Carlo simulation the video bitrate is the actual produced bitrate of the encoder. If the encoder rate is higher than the available bitrate, the delay will grow. Then a rate adaptation algorithm should adapt to the available bitrate. In the best case the one way delay will not exceed its limit and will also return to its minimum level after the congestion is solved by the algorithm.

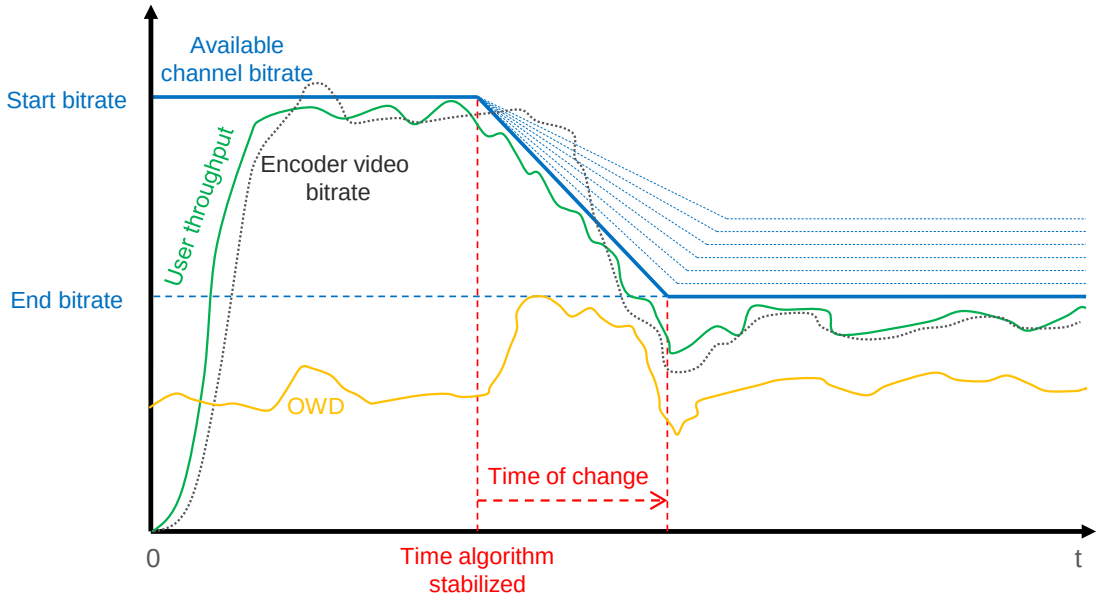

Figure 10 Monte Carlo method to evaluate algorithm reaction.

5.2.3 LTE Best Effort Bearer

The second and third scenario are LTE radio system simulations, which are a 3GPP Case 1-like scenario with 21 cells and 5 MHz bandwidth on a 2 GHz carrier frequency and SIMO antenna structure [33]. 5 MHz were taken due to the long time the simulation took.

Figure 11 gives an overview of the system simulation. The algorithm framework is embedded into the simulated mobile user equipment (UE). For each session a mobile user is connected with a fixed user on the network side, so that there is only one radio interface for uplink and downlink respectively. Only downlink simulations are performed so that the video is only send from fixed user to mobile user. However the feedback is send through the uplink channel of the network.

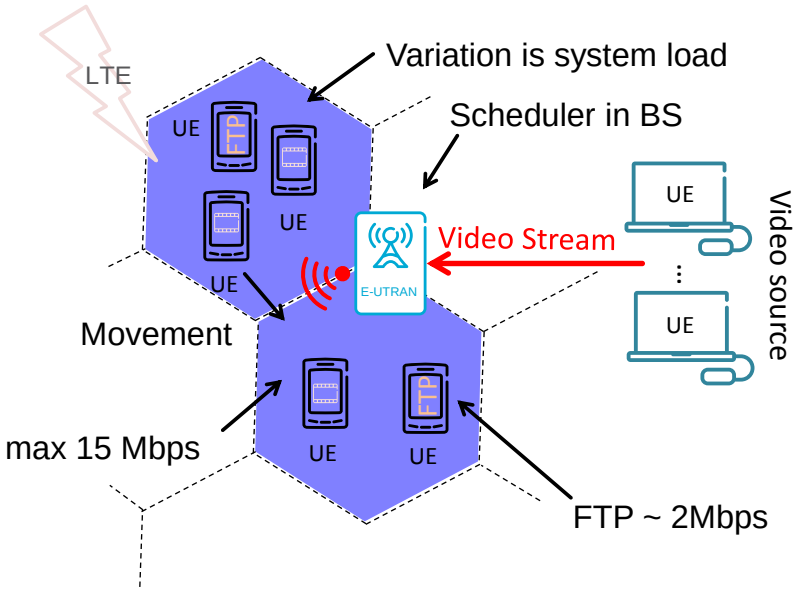

Figure 11 Downlink Setup including network components.

The video used for the simulations is a constant bitrate video with a nominal bitrate of 1.5 Mbps. The video source is rate adaptive between 0.15 to 1.5 Mbps and periodically provides frames with 30 fps. The length of each call is limited to 60 seconds. The actual output rate of the encoder may fluctuate around the nominal bitrate. The video media contains idle and data-limited periods like it is demanded in [3]. The encoder simulator takes a trace based video source and scales the frames in the manner the adaptation framework suggests it. This doesn't fully strike the reality. A more detailed simulation would also simulate the encoder in a more precise way.

The users are distributed randomly over the simulation area and move in straight lines with random directions and a speed of 3 km/h. A simplified handover algorithm with ideal measurements is used. The variation of available bitrate for each user is done by the number of users system wide so that with more users the cell will be more congested. The arrival of users follows a Poisson distribution, which is proportional to the length of the call so that in mean the number of users per cell is kept equal. Users, who transmit files over TCP, are added as background traffic. These users are called FTP users. Because the file size is rather short and FTP users arrive through a Poisson process the system is frequently congested. The average FTP load is 2 Mbps. A proportional fair scheduler allocates the bandwidth for each user. In scenario LTE best effort bearer the video and FTP users are on the same bearer with the same priority, meaning that both session types are scheduled over best effort bearers.

In a stochastic simulation the parameters of the channel is not fixed. They are stochastic distributed. To be sure that the difference between the algorithms is not originated from the stochastic simulation, the parameters for every simulation were fixed for the simulations. Then, different stochastic parameters were taken to average the statements of the simulations.

5.2.4 Bulk Transfer

The fourth scenario is called "bulk transfer". In this setup the data, which is sent by the algorithm, is random, not limited and periodic data from the video encoder. It is used to show the behavior of the Sprout algorithm when the data which can be transmitted is unlimited.

Chapter 6 Results and Analysis

This Chapter presents the results generated by the evaluation framework and described by the KPIs. A good adaptation framework for cellular wireless networks must overcome the following challenges [2], [3]:

- It must cope with dramatic temporal variations in link rates.
- It must avoid over-buffering and high OWD, but at the same time, if the rate can be increased, it must avoid under-utilization.
- It must be able to handle outages without over-buffering, cope with asymmetric outages, and recover gracefully afterwards.
- It must cope with different media behavior, i.e., the media should contain idle and data-limited periods. An example is varying amount of motion for video.
- It must survive in competition with session, which have the same rate adaptation and also handle competing sessions with different adaptation control.

The scenarios taken for the evaluation in this thesis are used to preform exactly these challenges. Table 4 in Section 5.2 gives an overview of the scenarios. They are used to be able to extract the KPI to benchmark the algorithms and to detect misbehavior. In the following sections a detailed evaluation is presented. Quantitative and qualitative simulations are used to show the behavior of the different algorithms. None of the algorithms turned out to be the best for the case conversational video in mobile radio. All show minor or major misbehavior, which could be solved by using techniques from the other algorithms.

6.1 Introduction into Result Discussion

The KPIs explained in Chapter 3 were implemented in this thesis to evaluate the algorithms using Matlab. Figure 12 shows simulation results, which give a quantitative overview of the algorithm in the scenario LTE best effort bearer. Since this scenario describes the most realistic use case, misbehavior is a criterion for exclusion of the algorithm. The KPI used for evaluation are further described in in Chapter 3. The results for the four adaptation algorithms and a reference case are plotted. The reference case has a fixed video bitrate of 150 kbps where the video bitrate was constant throughout the video session. It is called minimum reference. Each algorithm is evaluated in a different simulation so that the influence between the algorithms does not lead to wrong conclusions. A simulation, where this influence is discovered is presented in Section 6.8.

In Figure 12 the graph 98 %ile of packet and 90 %ile of user OWD shows the worst OWD experienced by 90 % of the users if 2% of the delayed or corrupted packets are not considered. This assumption is made, because a low OWD cannot be served to all users over the whole time by any algorithm. The minimum rate serves a low OWD until the system is loaded with a high amount of users. This is comprehensible since the minimum video rate is not utilizing the network, which can be seen in the utilization plot of Figure 12. The same conclusion can be made for Sprout. Sprout works fine with bulk transfer which is shown in Section 6.10. But in the conversational video simulation the video bitrate of Sprout is almost

equal to the minimum bitrate and does not utilize the network. The utilization is nearly proportional to the system load. So it only depends on the number of users in the network and not on the behavior of the algorithm. Since the utilization is low, the OWD is not critical for Sprout. An explanation is given in Section 6.10. Although the video rate of Sprout is nearly equal to the minimum rate it adds delay by queuing up packets because of false congestion detection.

However, GCC utilizes the network more than Sprout and the minimum reference case, there is still room for a higher utilization. The algorithm does not adapt to the available bitrate. GCC is very conservative at low load of the system and achieves a lower rate than E-Sprout and Self-Clocked. As the system load increases, many of the users using GCC have an unacceptable high OWD, although the system is not fully utilized. The amount of users which can be served with a GCC controlled media session is much lower than a session controlled by Self-Clocked. The packet loss rate is very low for GCC, compared to Self-Clocked or E-Sprout. Since the packets are only dropped by the network and only if the buffers are full the algorithm does not have a big issue with packet losses. As said in Section 5.1 the buffers of the scenario are rather large. An explanation for the misbehavior of GCC in too long OWD is given in Section 6.11.

The Self-Clocked algorithm preforms better in this scenario. It keeps the OWD low while utilizing the network completely. Also with a high amount of users the OWD is kept under the desired level. Self-Clocked and E-Sprout achieved the highest video bitrate. The packet loss is rather high for medium and high number of users in the system. However this KPI can lead to misunderstandings, since it can be better for the user experience to drop packet than being disturbed by a high OWD for a long period of time. An explanation of this high packet loss is given in Section 6.12.

E-Sprout shows that by adding the RTT as a metric and introducing the changes described in Section 4.5 to Sprout, it is possible for E-Sprout to keep up with the other algorithms. The behavior of E-Sprout on conversational video is much better than the original Sprout. However, E-Sprout is not at the same stage as GCC and Self-Clocked. The OWD is only acceptable for a low amount of users in the system and the packet loss is too high for medium and high amount. The video rate is acceptable, and the algorithm is utilizing the network in the same amount as Self-Clocked. E-Sprouts misbehavior is discussed in Section 6.4 and 6.5. In the same way like Self-Clocked, E-Sprout is preventing the channel from congestion by packet conservation and packet dropping if too many packets are queued up, because the amount of dropped packets is higher. However, compared to Self-Clocked the amount of dropped packets is too high.

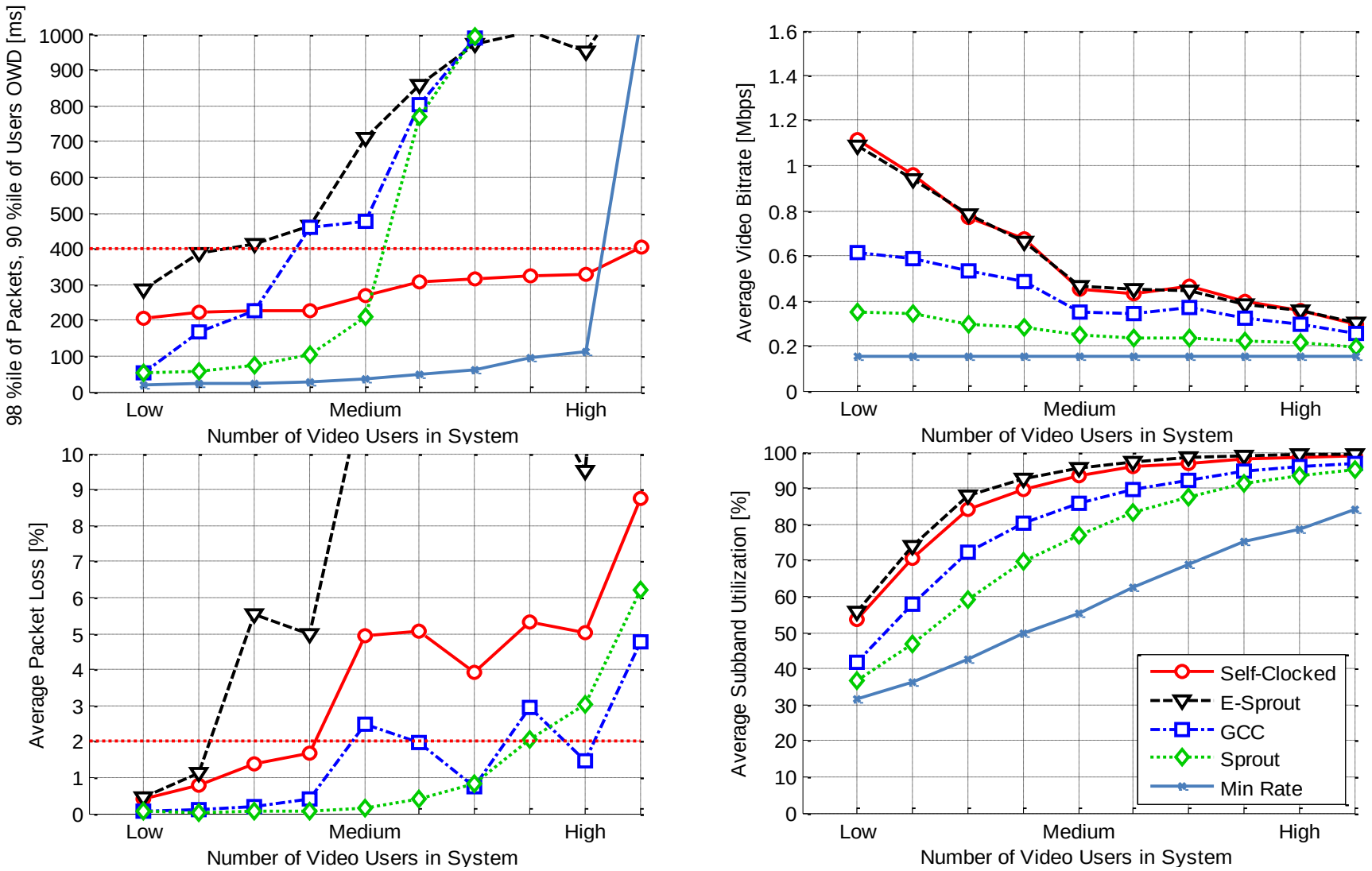

Figure 12 Simulation results for the different adaptation algorithms in LTE best effort scenario. A minimum fixed video bitrate of 150 kbps is used as a reference.

6.2 Overhead

The overhead of a video conversation is high compared to a pure audio conversation. Nevertheless, in comparison to the actual transmitted data the overhead timestamps and sequence numbers is negligible [34]. So, the amount of data added by Self-Clocked and GCC is per se not too much. The amount should be proportional to the available bitrate variations or the user throughput. If the channel is not changing the absence of control data should be noticeable. None of the algorithms behaves in such a way. In Figure 13 the overhead for each algorithm evaluated in the LTE best effort bearer scenario is calculated. The figure shows that the amount of overhead is quite low for all algorithms if the system contains a low number of users. If the system is more filled and the amount of congestions grows, the amount of overhead grows for all algorithms except from Self-Clocked. Since the algorithm has a fixed overhead size of 8 bytes per packet, the overhead of the algorithm is not sensible of available bitrate variations. GCC and E-Sprout will send more data, if the available bitrate changes more frequently. Since these algorithms send information of the channel in their feedback the amount of data is higher than for Self-Clocked. Self-Clocked only sends the timestamp and the amount of received data. E-Sprout compared to Sprout has a better overhead because Sprout has a much lower video bitrate, and thus the overhead data is of greater importance.

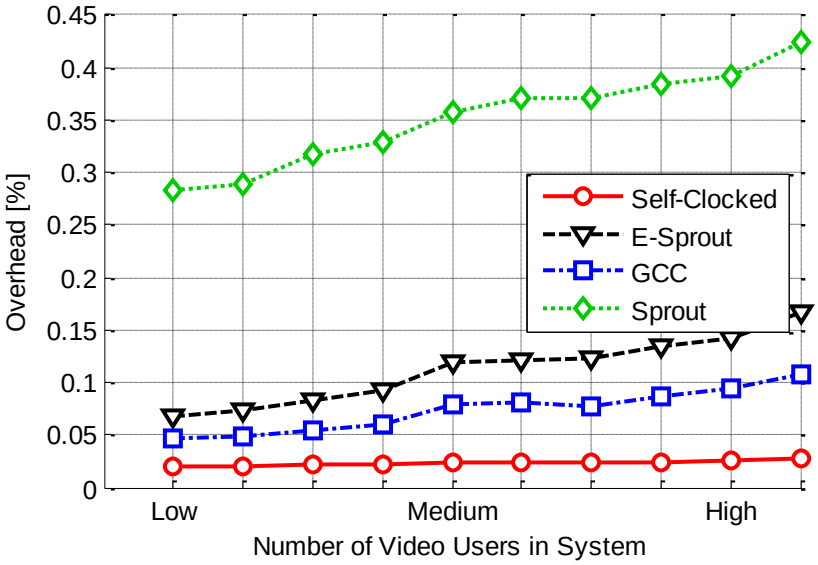

Figure 13 Overhead

6.3 Qualitative Explanation of Behavior of Algorithms

In Figure 14 and Figure 15 example graphs for the different algorithms are plotted. The green line marks the throughput of the user, blue marks the available bitrate in average, and black is the encoder bitrate. This bitrate is the real bitrate produced by the encoder and controlled by the algorithm through the target bitrate. The red line is the OWD measured from camera to screen. The packet losses are marked with the light blue color. This example graphs are qualitative explanations of behavior of the algorithm taken from the Monte Carlo simulations.

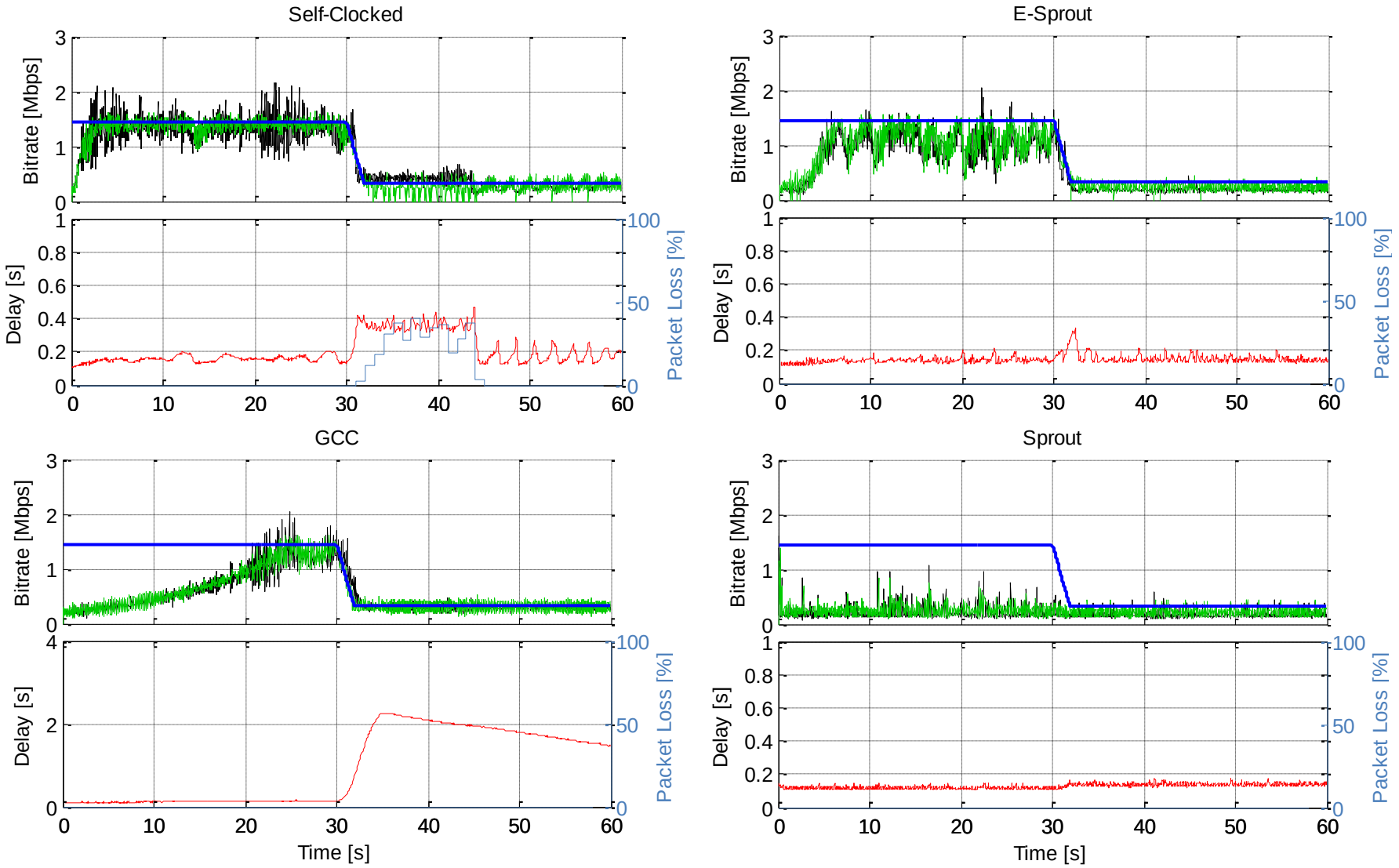

Figure 14 Start behavior and sudden occurrence of congestion. Available bitrate (dark blue) changes in 1.84 seconds from 1.43 to 0.32 Mbps. User throughput (green), encoder bitrate (black), packet loss rate (light blue) and OWD (red).

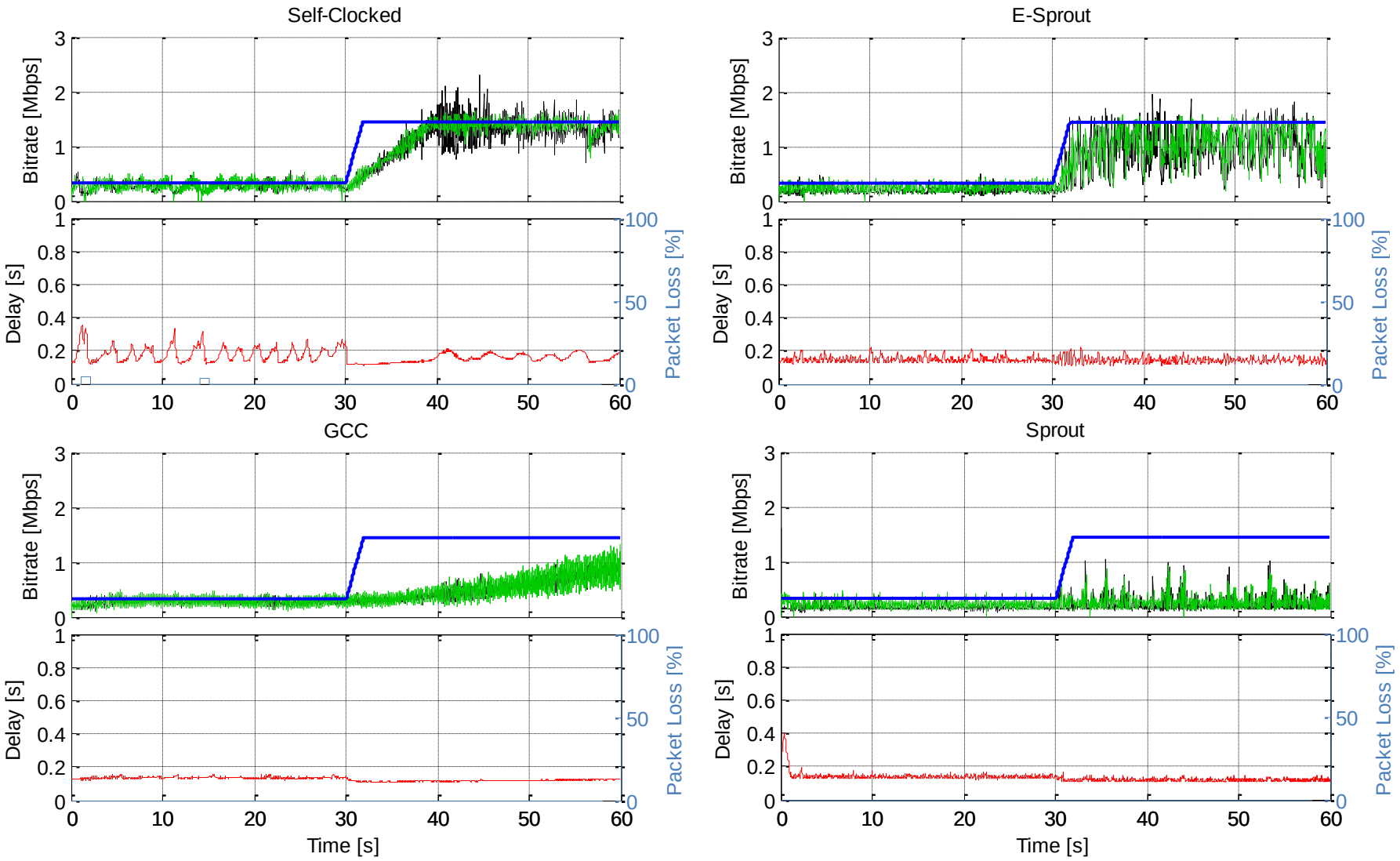

Figure 15 Sudden increase of available bitrate (dark blue) from 1.43 to 0.32 Mbps in 1.84 second. User throughput (green), encoder bitrate (black), packet loss rate (light blue) and OWD (red).

6.3.1 Start Behavior of Algorithms

In the first seconds of Figure 14 the startup behavior for each algorithm can be seen. The startup time for an algorithm is very important for the user experience. If the video has a bad quality for a longer time the user will be annoyed and maybe switch of the video before it is actually at its target bitrate. GCC has a very long start up time. The available bitrate is increased by a fixed factor and is not changing over time. Although this guarantees that the available bitrate of the channel or the maximum video bitrate is achieved, GCC needs a long time to stabilize the bitrate. Through its exponential start up Self-Clocked is reaching the available bitrate very fast. Also, E-Sprout does not displease the users with a long startup time. It waits for the second forecast to arrive and only after that starts to utilize the channel. The reason for the short startup time of E-Sprout is its ability to measure the channel condition directly. This is discussed in more detail in Section 6.15. Since Sprout is not able to utilize the channel, which is discussed in more detail in Section 6.10, a conclusion to its startup time cannot be given.

6.3.2 Reactivity to Sudden Congestion

In Figure 14 congestion is simulated after 30 seconds. The available bitrate of the channel is falling from 1.43 to 0.32 Mbps in 1.84 seconds. 30 seconds is needed to guarantee that the algorithms stabilize up to the available bitrate. Sprout is not utilizing the network, so that a conclusion for its congestion detection cannot be given.

The Self-Clocked algorithm is adapting to the available bitrate. In the time the congestion occurs the delay is growing directly, but Self-Clocked adapt very fast. The delay touches its limit but the algorithm manages to hold it on a proper level. However, the delay is kept low, the algorithm could do more. The delay which is consistent could be erased by sending low bitrate for a short time. Self-Clocked drops packets, if the age of the packet waiting to be send is too high. Figure 14 shows, Self-Clocked drops a lot of packets after congestion occur. The target bitrate of the encoder is still too high, although Self-Clocked measures bad channel conditions. This leads to the conclusion that the rate control is not sensitive enough for a sudden congestion. It should lower the bitrate, and prevent the packet from being dropt before transmission. Further details are given in Section 6.12.1.

E-Sprout reacts very fast to congestion. The delay is not even touching the limit of 400 ms. Additionally, the self-inflicted delay is erased after the congestion. This erasure is done by underutilizing the network, so that the channel is cleared of congestion again. The algorithm follows the available bitrate.

At first glance, GCC seems to react very similar to congestion like the other algorithm. However, there is a difference. Since GCC does not control the encoder output, the amount of bytes transferred into the network is high while congestion begins. Thus GCC adds a lot of self-inflicted delay and the receiver estimation is not sensitive enough to solve the congestion directly. In this simulation the buffer size of the network is very high. So, the principle of packet losses for measurement is not useful. The sender side rate control only takes the receiver estimated bitrate into account and does not have the information on packet loss. Thus, the decrease of the target bitrate in one evaluation step of the rate control

is only 5%, which is insufficient in order to directly solve the congestion and secure delay lower than 400 ms.

The packets, which provoke the self-inflicted delay, stay in the network. The time, until the delay adapts to an acceptable value, is much too high. The algorithm should send low video quality for some seconds to reduce the delay. In theory, this should be done with the under use state. However, the evaluation shows, that the estimated bitrate stays at the actual bitrate of the channel.

6.3.3 Reactivity to Available Bandwidth

Video users do not like to see many changes in quality during a session. A reduction in quality should be unfrequently. An upscale to a higher quality could always be very short in time. The algorithm should provide a fast upscale on the one hand, and on the other hand should not run into congestion too frequently. A complete answer in not given, however the evaluation shows, that E-Sprout runs into congestion too frequently, which is further discussed in Section 6.4.

If the available bitrate of the channel rises after congestion, an adaption algorithms should raise the encoder bitrate to give the maximum possible video quality to the user. Figure 15 shows the algorithm in a congestion scenario at startup. After second 30, the available bitrate of the channel raises from 0.32 to 1.43 Mbps in 1.84 seconds.

Self-Clocked and E-Sprout do perform well in this scenario. Self-Clocked is constantly probing the channel in the time of the congestion. The delay is varying in an acceptable interval. Self-Clocked is increasing the encoder bitrate from the moment the congestion is solved until the available bitrate is reached again. The plot also shows a disadvantage of the algorithm. Although the available bitrate is high at second 32 the algorithm increases the encoder bitrate with fixed factor. In this time the video quality is lower than necessary.

E-Sprout is also preforming well. The average OWD in the time of congestion is lower than the average OWD of Self-Clocked. The OWD of E-Sprout stays under the limit of 400 ms. When the congestion disappears, E-Sprout reacts very fast and adapts to the available bitrate, while not adding self-inflicted delay. However, the bitrate changes too frequently in small scale, which is a disadvantage of the algorithm.

After congestion, GCC needs long time to recover. This is due to the fact that the rate control increases the target bitrate with a nearly fixed factor η. It is nearly constant although the channel has a much higher available bitrate. The factor is inverse to the RTT, but the RTT is very low in the whole time of the call since the delay is kept small. So, it is problematical to take the increase factor as a function of the RTT. The estimated available bitrate gives a much better knowledge of the channel.

The original Sprout algorithm is not sensitive to the available bitrate. At the time of congestion, the video rate is equal to the minimum video rate. When the congestion disappears there are some spikes, where Sprout increases the bitrate. However these spikes are too small and too infrequent to fully utilize the channel.

Table 5 shows the maximal amount of throughput increase, called aggressiveness discussed in Section 3.2.3. The result is takes from the Monte Carlo simulation. This guarantees that the maximal aggressiveness is not depending on network conditions.

Algorithm	Self-Clocked	E-Sprout	GCC	Sprout
Aggressiveness	0.22 Mbps	0.85 Mbps	0.12 Mbps	0.31 Mbps

Table 5 Max aggressiveness over all Monte Carlo Simulations.

GCC has a very low aggressiveness as it is illustrated in Figure 15. The algorithm slowly adapts to the available bitrate. In the best case of the algorithm, which is not shown in the graph, GCC needs around 11 seconds to raise the throughput from the minimum video bitrate of 150 kbps to the maximum of 1.5 Mbps. Self-Clocked needs around 6 seconds to raise from minimum to maximum. This is better than GCC, but it still misses to use the full bandwidth of the channel.

A prompt improvement of the video quality is desirable, in order to satisfy the user. However, an algorithm should also ensure that the bitrate does not change too frequently, even though the network has some capacity left. If the time of a higher available bitrate is short the algorithm should neglect this. In this point E-Sprout has a draw back. The time to raise the throughput from minimum to maximum only takes 1.5 seconds in best case. However, it reacts on every change of the channel, which can be annoying for the user, if the channel changes very frequently. The algorithm is too aggressive.

Sprout has a relatively high aggressiveness for its low achieved bitrate. The reason is that Sprout is changing the target bitrate frequently between maximum and minimum. In average the encoder bitrate is low but there are spikes when Sprout tries to raise the bitrate in one or two frame periods. This is a very unstable behavior.

6.4 Responsiveness

The KPI responsiveness is taken to show the reactive behavior of the algorithms in the LTE best effort scenario. If the OWD is above 400 ms congestion is indicated. A low value of responsiveness indicates a fast adaptation and reactivity. Figure 16 shows, that the responsiveness of all algorithms end up in nearly the same value if the system is filled up with a high amount of user. However, the reactivity to congestion is different as shown in Section 6.3.2. The conclusion is that all algorithms manage to solve congestion in 50% of the time between two congestions. This result of the graph can lead to misunderstanding, since the amount of congestions is different between the flows of each algorithm. In the left graph of Figure 16 the amount of congestion is shown and it can be seen that E-Sprout and Self-Clocked run into congestion more frequently. Thus, the time between congestion is shorter. This means that the reactivity to congestion is much higher for E-Sprout and Self-Clocked than for Sprout and GCC. Since the GCC and Sprout are not utilizing the network, not congestion is triggered.

The reactivity of E-Sprout is very fast but it also runs into congestion too often. The value 0.25 of the average number of congestions per second shows that the algorithms suffers congestion every 4^{th} second. This leads to a very bad user experience. The frequency of

congestion for Self-Clocked is also very high when the system is loaded with a high amount of users. This leads to the fact that if congestion is permanent as in the case of GCC the graph will show a low number of congestion occurrence.

The responsiveness graph is only shows data when the algorithms are running into congestion, which is not the fact for GCC and Sprout with a low number of users in the system.

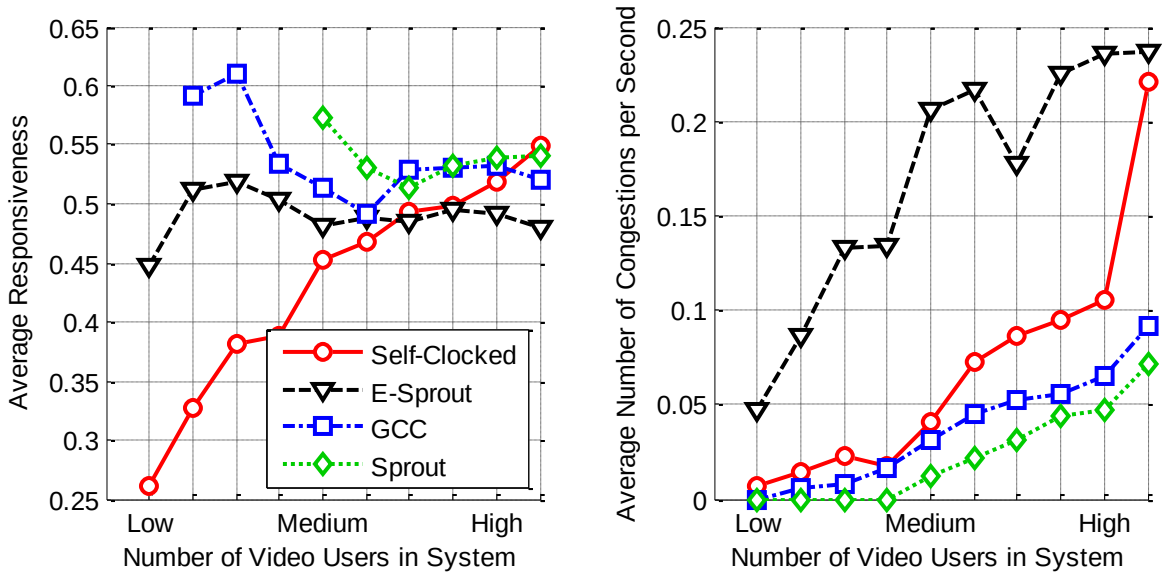

Figure 16 Responsiveness and number of congestions per second.

6.5 Stability of Algorithm

As mentioned in Section 3.2.1 instability has a critical influence on the algorithm performance and thus on the user experience. The coefficient of variation (CoV) of different metrics gives information about the stability of an algorithm. This thesis takes the CoV of the encoder bitrate. The advantage is that inner algorithm differences are not considered and the variation of the video itself can be subtracted. Since video data is usually varying, it impurities the evaluation. A stable algorithm delivering high video bitrate would not have a good CoV. In this evaluation the CoV of the pure video which has in average the same video bitrate, is subtracted from the CoV of the adapted video. Figure 17 shows the CoV for the different algorithms. The graph plots the average CoV over all users. For the evaluation the LTE best effort bearer scenario is taken.

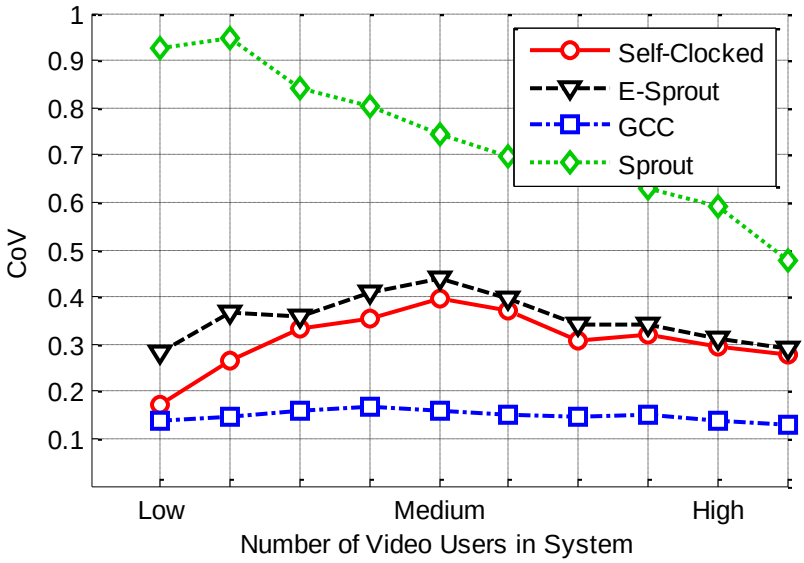

Figure 17 CoV of video bitrate over all users plotted over the system load which is varying with the amount of users in the scenario.

Except from Sprout all the algorithms have a lower CoV if the System is low loaded and high loaded. In between the CoV is higher. Figure 17 shows, that Sprout is changing the video bitrate too frequently. A CoV of 0.9 without occurrence of network congestion is unacceptable. The reason for this frequent variation is given in Section 6.10. Sprout is very unstable in use with conversational video. GCC has the lowest CoV. The OWD graph of Figure 12 shows that GCC is not fast enough in adaptation and misses to ensure the OWD to be under 400 ms. Obviously, GCC does not adapt fast enough. Due to its constancy, which is indicated by the CoV, it fails to cope with the frequently changing channel conditions. In Section 6.11 the explanation for this misbehavior is given.

Self-Clockeds CoV is higher. In the example graphs in Figure 14 it can be seen that it adapts faster to the available bitrate, although not exactly enough. It is not able to stop packet dropping which is further discussed in Section 6.12.1. However the OWD graph in Figure 12 shows that the adaptation works fine. So, Self-Clockeds CoV seems to be nearer to the optimum than the others.

The CoV for E-Sprout is much lower than for Sprout. This is achieved by applying the filtering of *BytesToSend*. As intended, the amount and the ratio of changes was reduced, compared to Sprout. Although the stability of the algorithm was improved, it is still too unstable. Due to the fact that E-Sprout drops packets, if it detects congestion, the algorithm has a high packet loss rate because of it aggressiveness and instability. It runs into congestion to frequently. In Figure 14 in the graph for E-Sprout at second 30 it can be seen, that the algorithm copes with congestion very well, but it is also noticeable that the algorithm is not preforming well in a stable channel. The encoder bitrate is much more varying compared to GCC or Self-Clocked.

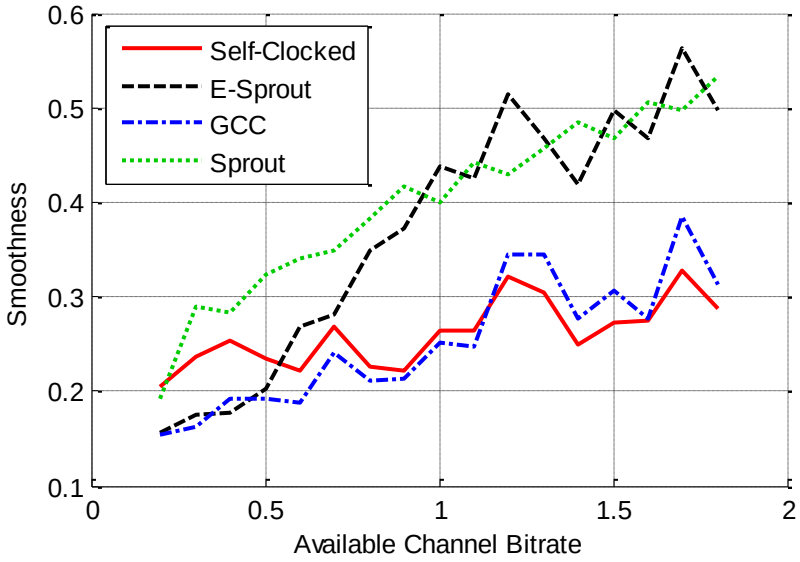

Figure 18 Smoothness made by Monte Carlo with constant available bitrate.

This can also be seen in Figure 18. This graph shows the smoothness of the user throughput in a steady state scenario. Monte Carlo simulations with different available channel bitrates were done to show the differences of the smoothness of the algorithms over the available bitrate. The available bitrate was fixed in every simulation so the algorithm is preforming in a steady state simulation. The figure shows that E-Sprout has a very high smoothness factor which is problematic since a smoothness factor of zero leads to good smoothness. Also, the smoothness of the algorithms is getting worse when the available bitrate is growing. This is comprehensible since the algorithms have more opportunity to change the user throughput. Even though, GCC and Self-Clocked have the best smoothness [17] argues that it is not enough. It claims that the CoV, which in this case is equal to the smoothness, should not be higher than 0.1 in a steady state scenarios.

For Self-Clocked it can be seen that although it has the same smoothness as GCC it reacts faster to available bitrate changes. Since Sprout and E-Sprout are very unstable the smoothness factor is very high which points out a very bad smoothness.

6.5.1 Informative Value of Coefficient of Variation

The CoV is useful to compare the algorithms but it does not give an absolute answer to the stability of the algorithm. On the one hand an algorithm should adapt fast enough to follow the available bitrate, on the other hand, the algorithm should not be unstable through false congestion detection. A preferred range of CoV cannot be given. [17] says, that a CoV of 0.1 leads to a high instability. This applies for fixed internet but it does not for highly varying mobile radio channels as the channel would be under or overused frequently.

6.6 System Stability and Utilization

The stability of the system is an issue for any mobile network operator. If an algorithm causes instability for the system throughput, all users will suffer from that. The available throughput of the users will change frequently, not because of mobility or other radio

problems but because another instable algorithm varies in resource allocation. Figure 19 shows the CoV of the system throughput depending on the number of users in the system. The system gets more stable as there are more users. Thus also the algorithms get more stable like it is said in Section 6.5. The instability, when the system is less loaded, is evoked to the variation of the video bitrate and is not generated by the algorithms. This is seen by the variation of the system throughput of the minimum bitrate reference. Interestingly, GCC, which is the most stable algorithm, evokes a higher instability than E-Sprout and Self-Clocked. The reason is that this graph is not plotting the stability over the utilization. Since E-Sprout and Self-Clocked utilize the system more than the others the stability of the system is higher for these algorithms. Also minimum bitrate reference and Sprout do not utilize the system as much as E-Sprout and Self-Clocked, so the system stability is also low in their case. As a conclusion it can be said that the system stability is rather unaffected from the instability of the investigated algorithms. It is much more sensitive to the system load.

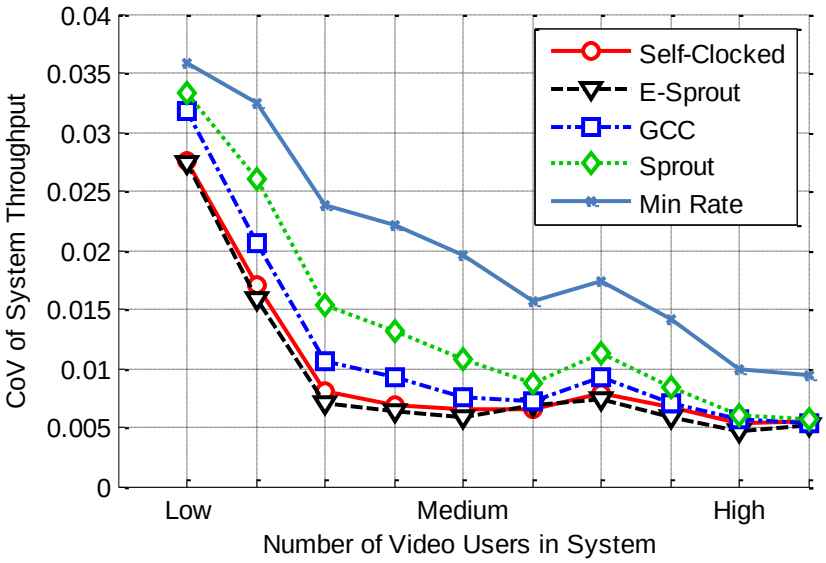

Figure 19 System Stablility.

For the utilization the spectral efficiency can be used (see Figure 20). However, it gives the same conclusion as the subband utilization. The more subbands are used and the more the system is utilized the higher the spectral efficiency is. Since the subbands are used more efficiently, the algorithms influence the efficiency of the system. However this graph looks good for E-Sprout one has to be careful with this statement. An algorithm which was not sensitive to congestion and would inconsiderately transmit data without any guarding structure would have the best spectrum efficiency. In the case of conversational video, where a too frequently congested channel will lead to disappointing user experience, a more conservative algorithm would lead to a good solution.

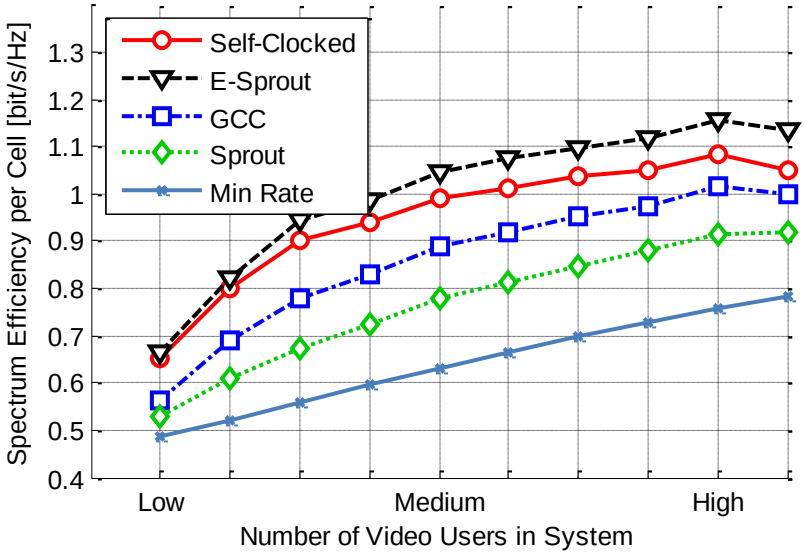

Figure 20 Spectral Efficiency.

6.7 Fairness between Users

The fairness between users using the same rate adaptation framework is evaluated in the bitpipe scenario with fixed available bitrate. The LTE best effort scenario is not taken, since the influence of the scheduler and channel changes depending on mobility falsifies a clear statement. The average amount of users sharing the same bottleneck is 10. The total available bitrate of the bitpipe changes so the fairness can be evaluated for different channels characteristics. Figure 21 shows the fairness index over the average bitrate per user. The average bitrate per user is calculated by dividing the total available bitrate of the bitpipe through the amount of users. The fairness index for all algorithms is rather high, which leads to a good fairness. Only the users using the Self-Clocked algorithm for rate adaptation suffer from a small unfairness among each other. The fairness is nearly independent from the available throughput per user.

Figure 22 shows Ɛ-Fairness in the same setup as taken for the fairness index. The Ɛ-Fairness shows the fairness between the user having the best throughput and the user suffering the most. In comparison to the fairness index in Figure 21 the Ɛ-Fairness is inversely proportional to the fairness. A low Ɛ-Fairness factor leads to a high fairness. The Ɛ-Fairness is rather more critical. It shows that for the Self-Clocked algorithms the user suffering the most is much more suppressed than by using another algorithm. Only with high available bitrate the Ɛ-Fairness factor for Self-Clocked is lower. This is comprehensible since the user can use more bitrate without supplant another user. The same is essential for GCC with high available bitrate. E-Sprout and Sprout have problems in this case. The aggregated throughput for the different users is not equal, although the system is not fully utilized. Nevertheless, Self-Clocked has the biggest problems with Ɛ-Fairness when the users have to share the bottleneck.

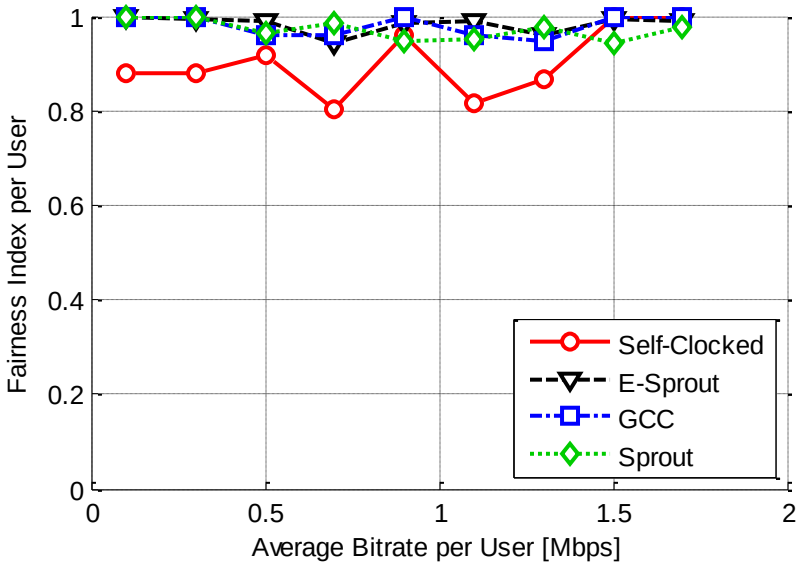

Figure 21 Fairness Index.

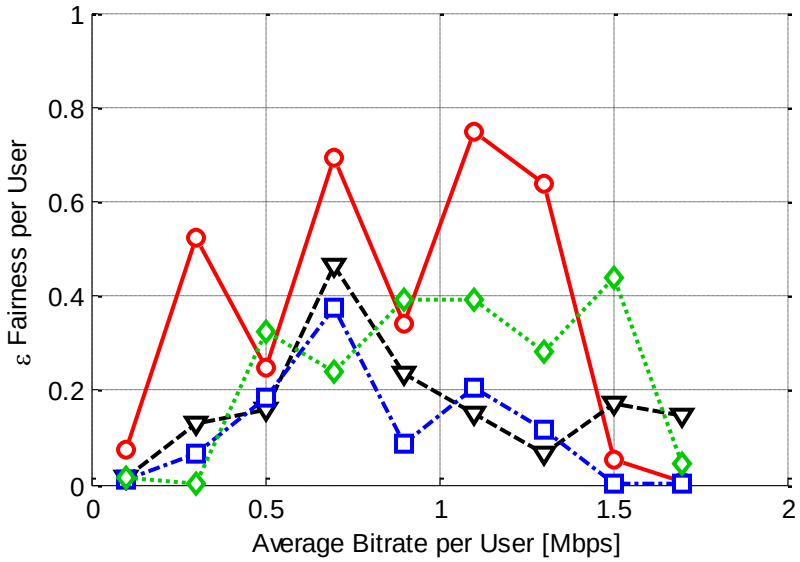

Figure 22 Ɛ-Fairness (Min/max) (Strong influence of minimum user).

The fairly shared spectrum efficiency evaluated in the LTE best effort bearer scenario is shown in Figure 23. It is comprehensible that the amount of fair shared spectrum is very high, if the system is only used by view users, since they share the same bottleneck but it is not congested. Since Sprout users are only receiving a low video bitrate there is no bottleneck. The amount of FSSE stays nearly the same, although more users are added to the system. The same applies for the minimal bitrate reference. For GCC and Self-Clocked the user suffering the most from the others have the same FSSE as the minimal bitrate users. This which means, that this user is fully suppressed to the minimum bitrate. E-Sprouts users have the lowest FSSE. Consequently, there is a user whose throughput is nearly at zero and no video is transmitted at all. Although Self-Clocked is more unfair then E-Sprout the minimum user gets more bitrate by using Self-Clocked than E-Sprout. This means, that the difference is higher for Self-Clocked but the minimum is also higher. E-Sprout seems to

have more users at a common level but simultaneously cuts the bitrate for the neglected users. This is a draw-back of the algorithm.

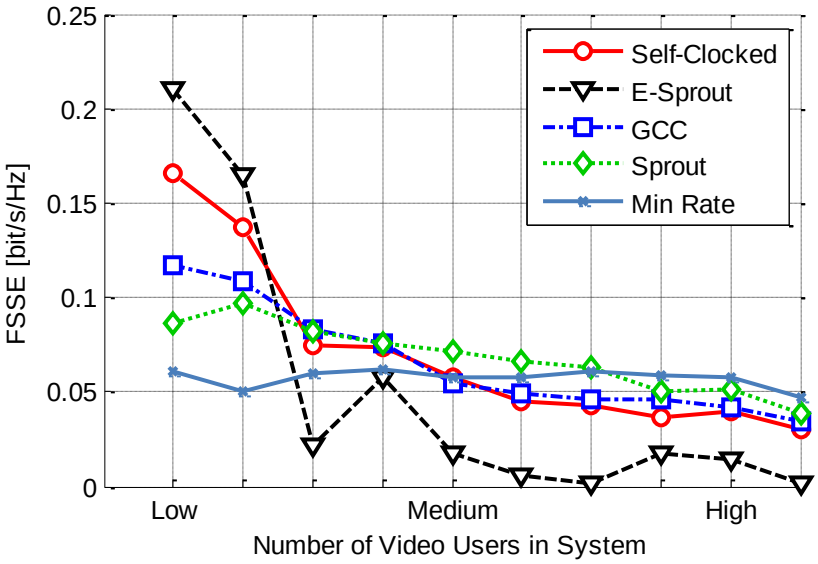

Figure 23 Fairly shared spectrum efficiency.

6.8 Fairness between Algorithms

Figure 24 shows how a TCP session suffers from a session with the different algorithms. This graph was evaluated using the LTE best effort bearer scenario. The average bitrate of the FTP users is very high when the number of video users in the system is low. However, the average FTP rate should be 2 Mbps it can be seen that it sometimes is higher than the average. The average bitrate of the system is meant over the whole time of the simulation. In this graph the average bitrate is estimated through measuring the average throughput of each FTP transmission. Since these transmissions are not limited the throughput can be much higher than the average over the whole simulation. If the bitrate of each TCP transmission is higher than 2 Mbps there are also times where no FTP user is transmitting anything. If the system gets more filled with video users the average file transmission rate will go down, but the amount of data which has to be transmitted stays the same. By a comparison with the average bitrate of the video user it can be seen that the media rate adaptation algorithm suffer the most. The TCP algorithm is less sensitive to congestion and is not adapting so strongly as the investigated algorithms. This is comprehensible, since there is no packet loss in this evaluation, because the TCP bitrate is sensitive to the RTT and packet loss, as shown in Section 3.3.4 in the TCP fairness KPI. In conclusion, the investigated algorithms are TPC fair. The difference between the algorithms is caused by the different utilization of the system by the algorithms.

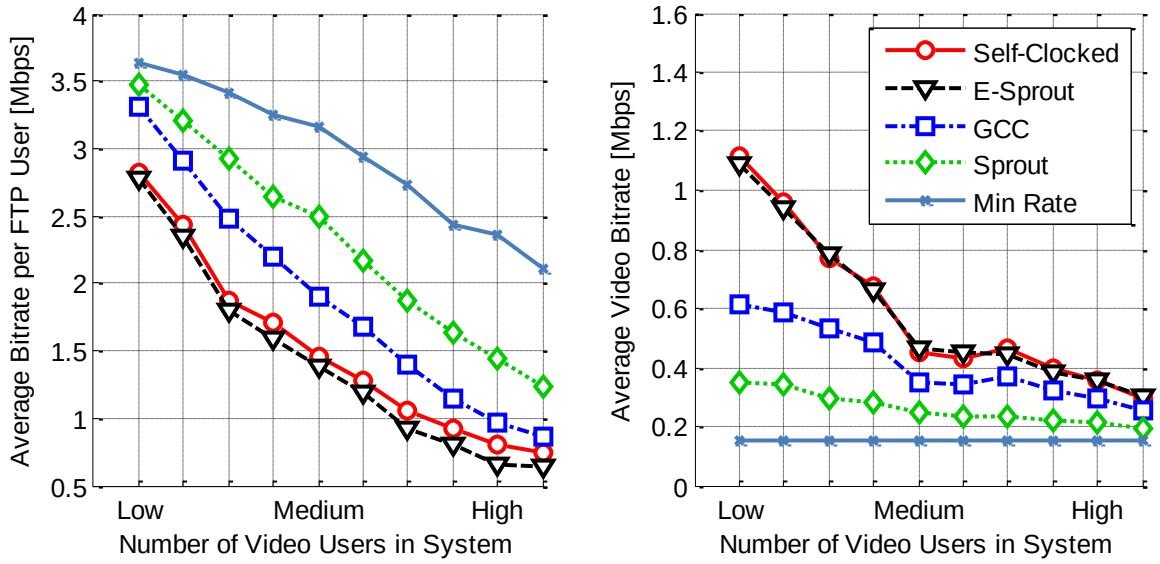

Figure 24 Fairness to TCP sessions.

Figure 25 shows the bitpipe scenario with all algorithms sharing the same bottleneck. The graphs show that the Self-Clocked algorithm is suppressing all the other algorithms. The graph is created by evaluating different simulation with fixed available bitrate but by iterating the average number of users for each algorithm. When the system is not in high utilization the algorithms do not suffer from each other. In higher load the amount of utilization growth and an influence between the algorithms is noticeable. The Self-Clocked algorithm is not the most aggressive one as shown in Section 6.3.3. An answer why it suppresses the other algorithm cannot be given in this thesis. An idea is that a high aggressiveness compared with high stability leads to the suppression of other sessions. This educated guess could be evaluated in further studies.

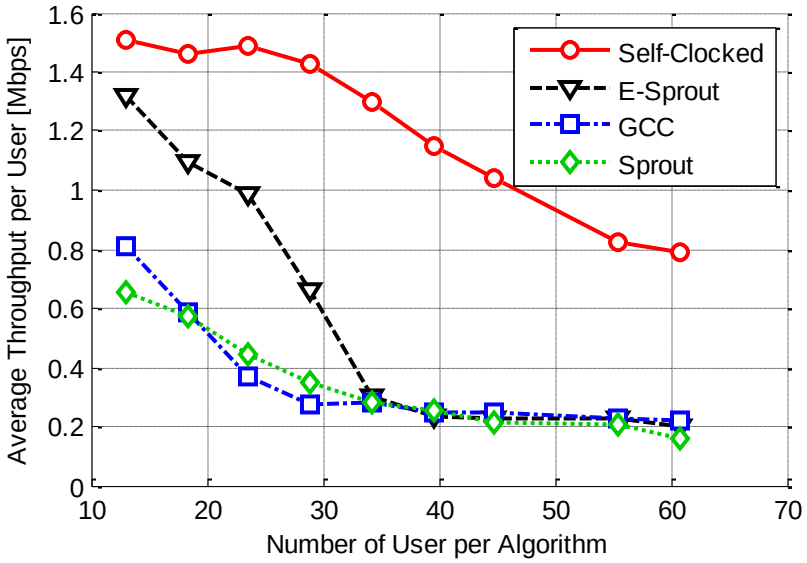

Figure 25 Fairness between Algorithms.

6.9 Monte Carlo Results

As an innovation this thesis is taking Monte Carlo simulation to create an evaluation framework. The algorithm is alone in the simulation and the available bitrate is varying between 2 and 0.2 Mbps. 2000 simulation where taken for each algorithm to get a good distribution of the alteration of each parameter. The parameters are the start bitrate, the end bitrate and the time of change. The time of change is varying between 0 and 5 seconds. The setup is further discussed in Section 5.2.1.

OWD Evaluation

The evaluation is done for the OWD and the packet loss rate. Since the three parameters and the amount of output (OWD or packet loss) form a 4 dimensional room, it was not possible to plot them in a normal graph. Also the possibility to mark the output with colors leads to a rather confusing picture. For that reason the output was filtered. The points in the graph indicate that for the set of those three parameters the once in the session the OWD is higher than 400 ms. Thus, a point cloud is generated which marks the volume where the investigated algorithm does not perform well depending on the parameters. Figure 27 shows this point cloud for each algorithm over the three parameters. This point cloud marks a volume. In a second step a surface is shaped as a border of this volume. The greater this volume is the worst is the behavior of the algorithm. The color of this surface has no meaning. It only helps the beholder to see the 3 dimensional structures. This volume is called “trouble volume” and is shown in Figure 26.

No algorithm can handle a high start and low end bitrate in a short time. GCC has the biggest trouble volume. It does not depend on the time of change which leads to the conclusion that GCC cannot handle a big change of the available bitrate neither it is abrupt nor over a distance of 5 seconds. Also, it is inconvenient for the user that GCC is not able to adapt from 2 Mbps to 0.9 Mbps without affecting the OWD to be higher than 400 ms.

The Self-Clocked algorithm and E-Sprout have the ability to handle big steps in an appropriate time. If the channel is not changing to quickly, the algorithm can follow the available bitrate no matter how big the difference is. The volume of Self-Clocked is a little bit smaller than of E-Sprout, but the algorithm has bigger problems with sudden changes from very high bitrate to medium low bitrate. An adaptation from 2 to 0.75 Mbps cannot be made at once. The point cloud in Figure 27 of the evaluation of Self-Clocked shows that there is a gap in trouble volume. Self-Clocked has no OWD problem when the end bitrate is over 0.5 Mbps and the time of change is higher than 1 second. E-Sprout is performing well in the Monte Carlo simulation. The Step size which can be overcome by E-Sprout is higher than all other algorithm, because of fast reaction. A sudden available bitrate change from 2 Mbps to 0.6 Mbps is possible. There are three outstanding points off the point cloud of E-Sprout which can be found by assuming a low start and end bitrate. By neglecting these points the trouble volume of E-Sprout would even be smaller than the one of Self-Clocked. E-Sprout has the ability to handle a channel change from 2 to 0.2 Mbps in 4 seconds which is better than the behavior of Self-Clocked and GCC.

The delay of Sprout is always low because the channel is never utilized, except of very low channel bitrate. It can be seen that Sprout has problems with low bitrate, although it should only send minimum bitrate of 0.15 Mbps. An explanation is given in Section 6.13.

Packet Loss Evaluation

For packet loss evaluation, a point is set if the simulation generates a packet loss rate higher than 2% and lasting for over 3 seconds. If there is more than one period of time with packet loss the time is summed up. So, it will also be considered if an algorithm loses packets frequently. Since Sprout and GCC do not drop packets and the buffer of the network is never running full, both algorithms do not generate packet loss in this simulation.

In Figure 28 it can be seen that the trouble volume of Self-Clocked is very large. To drop packets longer than 3 seconds is very probable. Self-Clocked drops packets if a low start and end bitrate is used. This should be avoided, since the algorithm should only send minimal bitrate video. The algorithm should only drop packets, if the minimal bitrate is send. So, when congestion is detected and too many packets are queued up before transmission, the algorithm should firstly use the minimal bitrate before dropping packets. A packet drop should always be the last solution. Also a packet loss over a time of 3 seconds should be avoided. When the algorithm drops all queued packets and sends minimal bitrate, the congestion should be solved. In this situation Self-Clocked drops still packets which make no sense.

By looking at the point cloud of E-Sprout it can be seen, that there are only view settings in which the packet loss is dramatic. This leads to a good user experience. However, the high amount of lost data in the packet loss rate graph in Figure 12, give another picture. There seems to be a conflict. The answer is that in the Monte Carlo simulations the available bitrate is constant except from the change at second 30. In a multi user multi cell scenario like the LTE best effort bearer the available bitrate varies much more frequently and instability of the algorithm leads to false congestion detection and for E-Sprout it leads to false packet dropping. A future task would be to make the algorithm more stable to abolish false congestion detection.

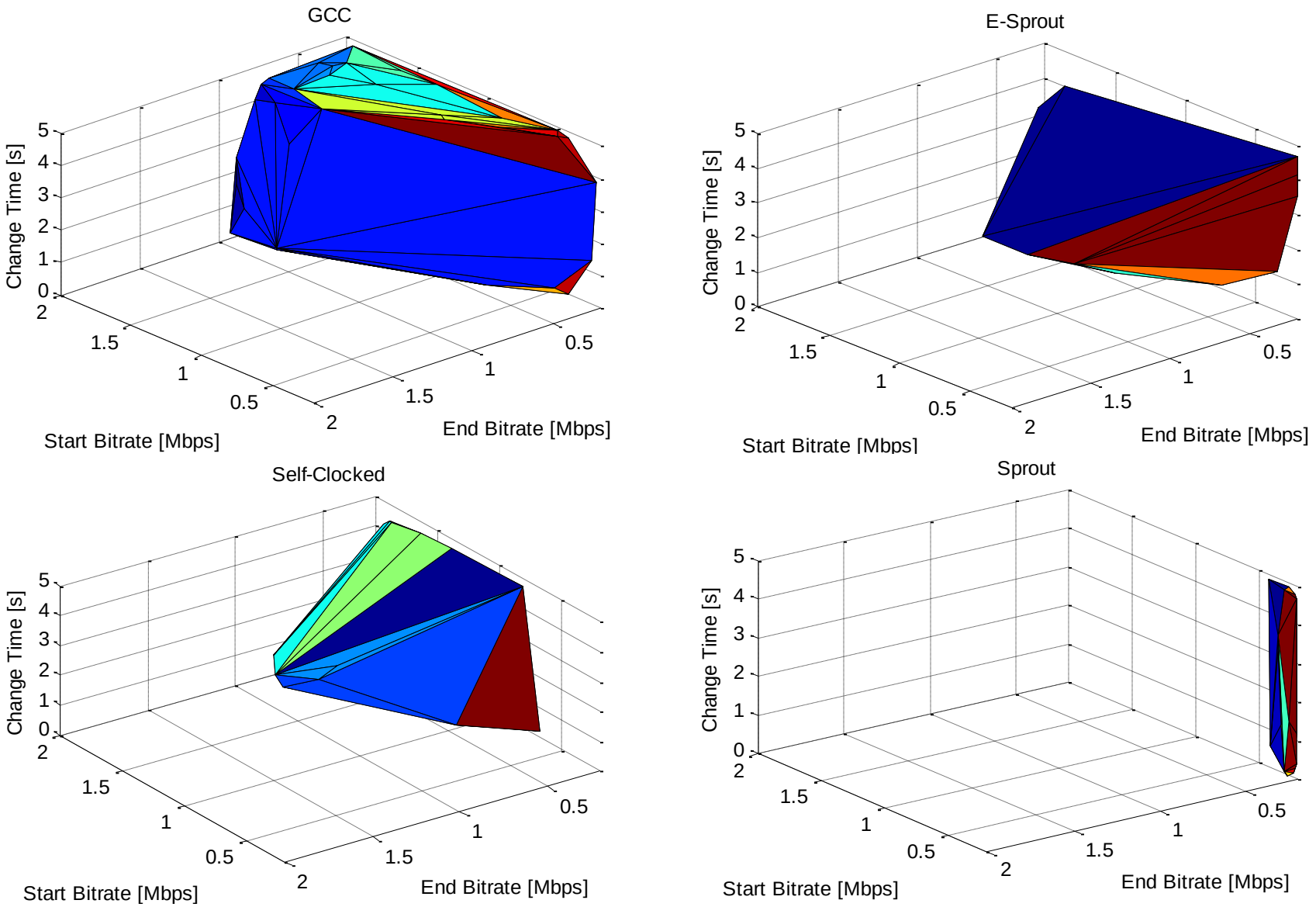

Figure 26 Monte Carlo Results Delay > 0.4s

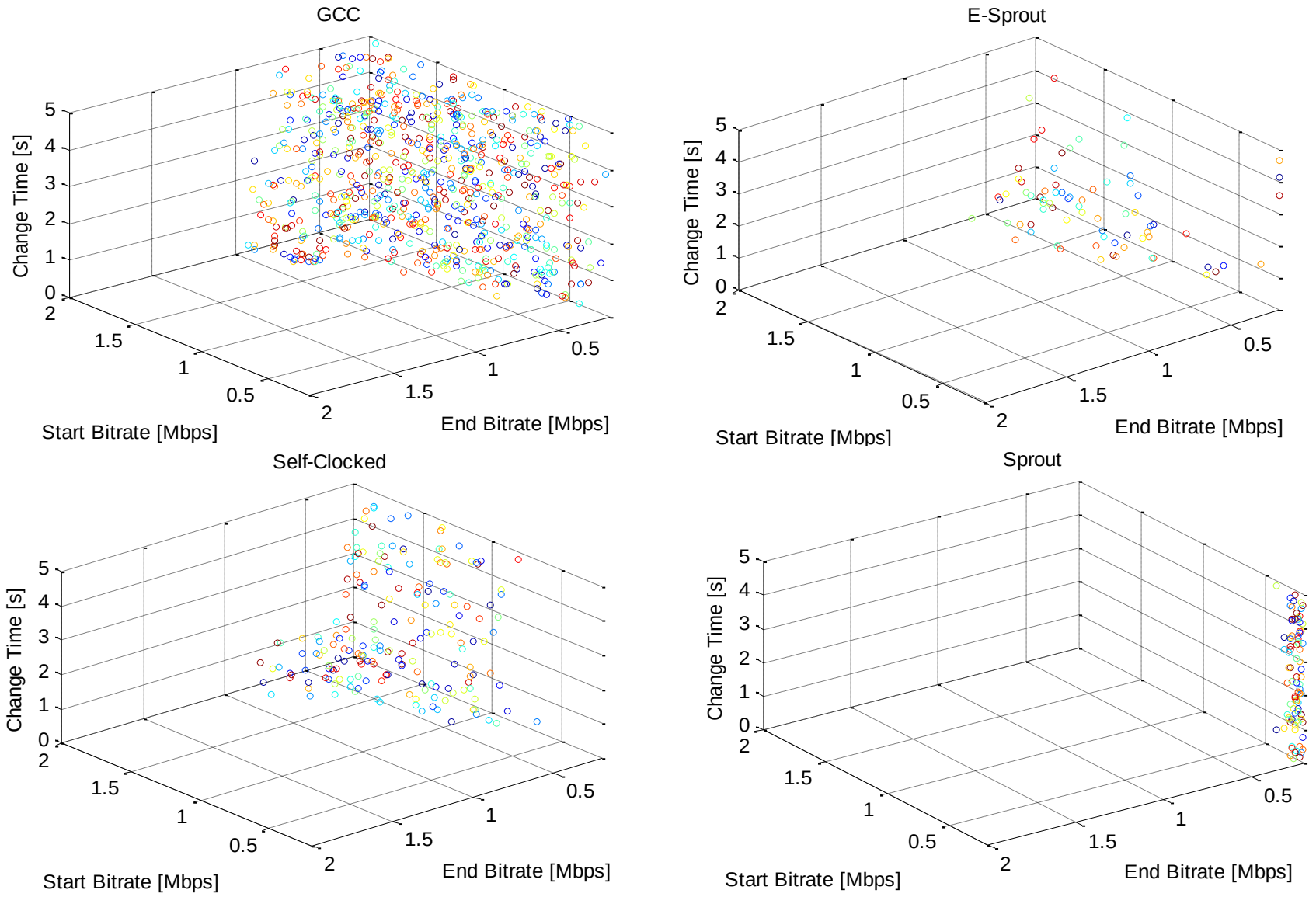

Figure 27 Point cloud delay >0.4 seconds

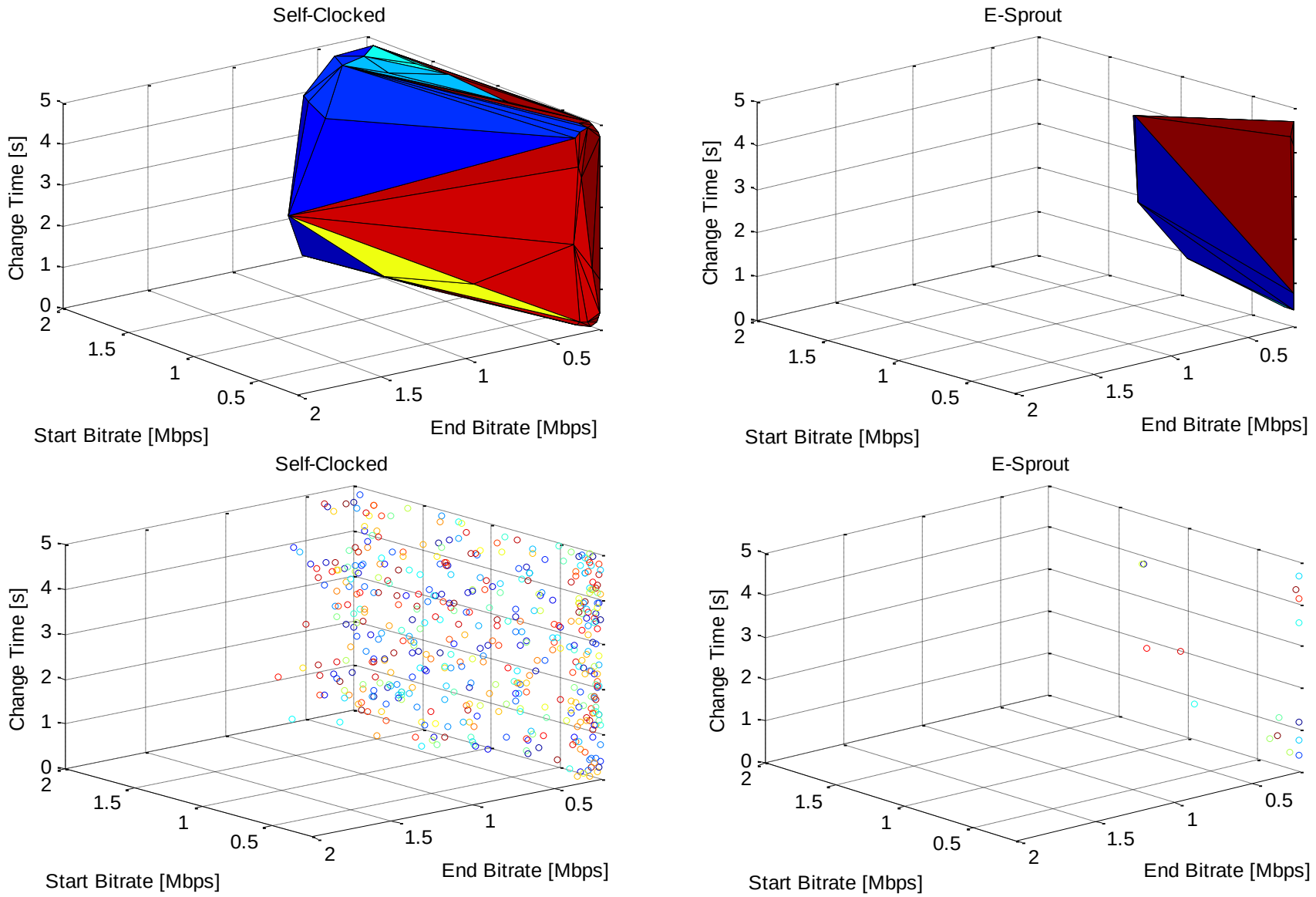

Figure 28 Packet loss rate > 2% and longer than 3 seconds on complete video time.

6.10 Sprout and Conversational Video

Figure 29 shows the inner algorithm metrics of Sprout at the beginning of a call in the scenario bitpipe, where the algorithm works in an uncongested channel. In the upper graph the data which can be sent is limited by the encoder. In the bottom graph the data is unlimited, like it is described by the bulk transfer scenario.

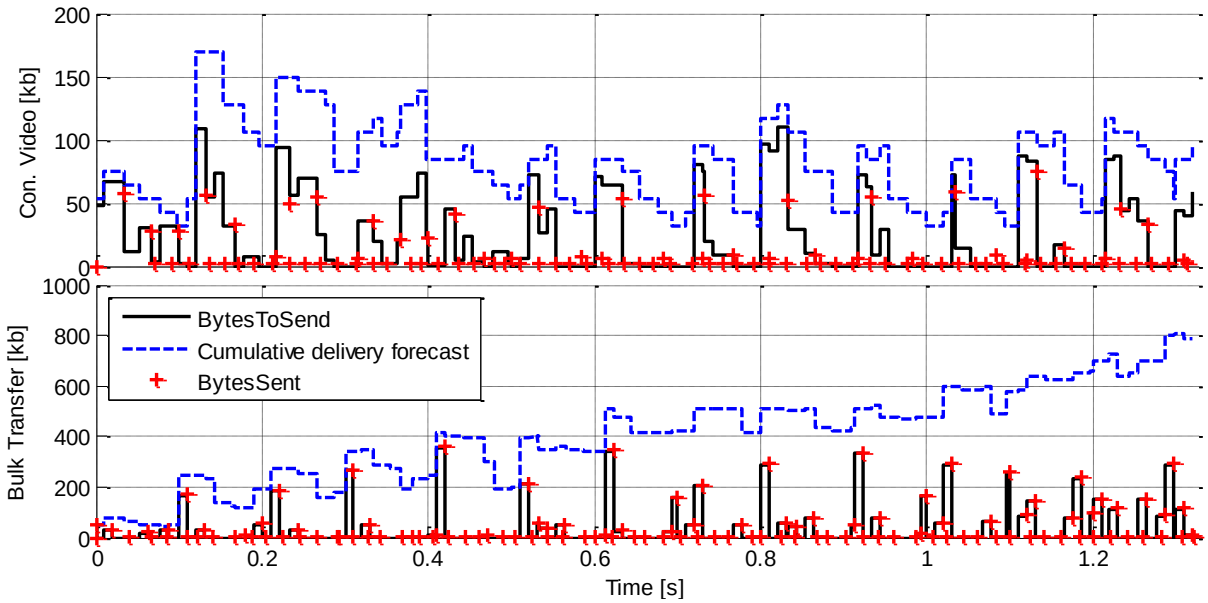

Figure 29 Cutout of Sprout's inner algorithm metrics. *BytesSent* is the amount of data actually transmitted. *Cumulative delivery forecast* is the channel estimation. *BytesToSend* is the amount of data save to transmit in this tick.

In the graph with the limited data it can be seen, that the amount of packets produced by the encoder is limited and periodic. *BytesToSend* and *BytesSent*, which is the actually amount of data transmitted to the radio link, are different. Therefore, the receiver measures the limited video bitrate instead of the available channel bitrate and detects false congestion. Since Sprout is only measuring the incoming throughput at the receiver, the measured signal is disturbed by the video stream. A video can vary strongly in matters of bitrate. If this is amplified by the algorithm, the result will be an unstable behavior which causes a very high CoV of the sending rate for Sprout (see Figure 17).

In lower graph with the unlimited data in Figure 29 *BytesToSend* and *BytesSent* are equal. The amount of measured data is equal to the available bitrate of the channel. Thus Sprout is more stable. Sprout satisfactorily utilizes the network with this setup.

Another thing which can be seen from Figure 29 is that the amount of unnecessary data is quite high for Sprout. It sends data every 50 ms although there is nothing to send or the amount of data to be transmitted is very small. This leads to a higher overhead for Sprout (see Figure 13). However this overhead is negligible compared to the video data.

6.11 Inertia of Google Congestion Control Algorithm

In the simulation GCC shows a conservativeness which leads to high OWD over a long time and user dissatisfaction. The problem comes from the way GCC adapts to the available bitrate. In the investigation it was seen that if a new feedback form the receiver arrives at the sender, the new bitrate will directly be taken as the target bitrate. So, the problem is that the estimation of the receiver side control is to optimistic and to slow in adaptation for the evaluated scenario. If the receiver estimates congestion because the output m of the Kalman filter is growing, it will trigger the overuse-state. The remote rate control will trigger the decrease state. In the decrease state the estimated bitrate $\hat{A}$ will be decrease with a fixed factor α, which is 0.95 because the rate control is in the decrease state. So, the estimated bitrate will decrease about 5% for every evaluation step. The investigation showed that this is not fast enough to handle rapid available bitrate changes.

GCC was developed to react very fast on packet losses. Even though packet loss is an common indicator for congestion, it cannot be taken as a metric of the channel in the evaluation of this thesis. Since the buffers of the network are rather large and a packet loss will only be visible after the network queues are filled up with such a high amount of data, that the accumulated data produces a very high OWD. The reason for this assumption was discussed in Section 5.1. In a further step an evaluation with active queue management or small network buffers could be done. However, the algorithm should also work without packet loss as a metric.

6.12 The Dropping of Packets on Sender Side to Keep OWD Low

When the channel is congested very strongly and there is a huge amount of data waiting to be delivered the depletion of the self-inflicted delay would need a long time since the data would have to be transmitted completely. Self-Clocked and E-Sprout are dropping packets, in such a case. Thus, the encoder has to start with a new I-frame once again, but the OWD will not be high for unacceptable amount of time. E-Sprout drops all packets and starts with the minimum bitrate. This guarantees that the network queues are emptied in a very short time. The user notices a short frozen picture or a black screen and after that the video is presented in lower quality. The other option would be a very long unsynchronized video, which leads to an unpleasant user experience. Figure 30 presents a qualitative explanation for this behavior. It shows a very strong change in the available bitrate of the channel. The red line marks the delay, light blue is packet loss rate, green is throughput, black is encoder rate and dark blue is available bitrate of the channel. Self-Clocked and E-Sprout drop packets at second 30. E-Sprout does this as well. For a short time it drops a huge amount of packets and then starts with a low bitrate. Self-Clocked is not able to stop the dropping which can be seen in Figure 14. The reason of this is further discussed in Section 6.12.1. The graph of GCC in Figure 30 shows the other solution. The network queues are filled with data and the depletion of the self-inflicted delay needs a very long time.

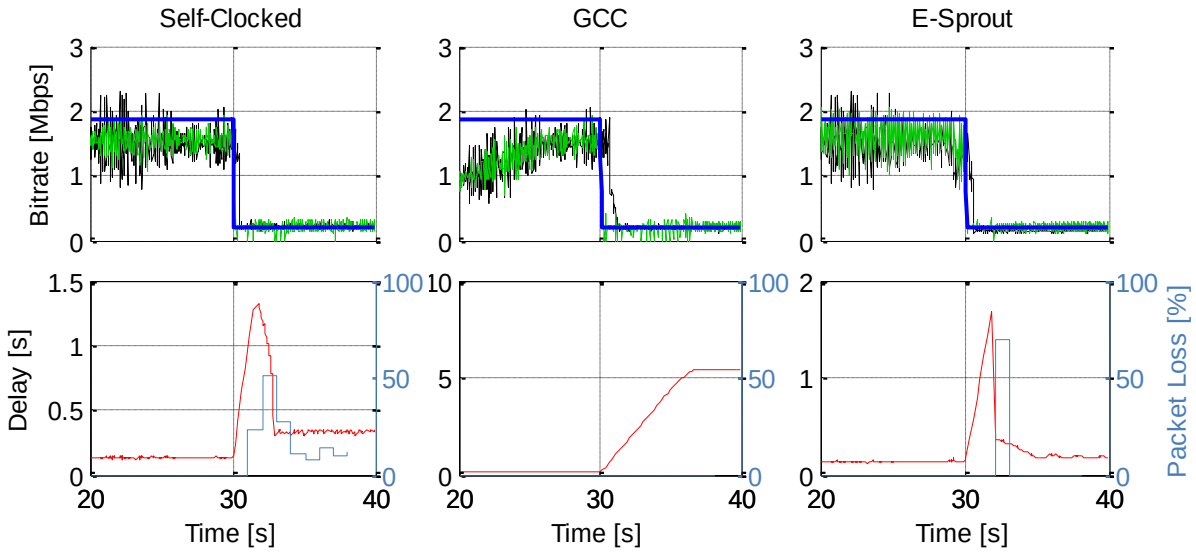

Figure 30 Sudden fall of available bitrate(dark blue) from 1.8 to 0.22 Mbps in 0.1 seconds. Throughput (green), encoder bitrate (black), packet loss rate (light blue) and OWD (red).

6.12.1 Unnecessary Packet Dropping of Self-Clocked after Congestion Occurrence

As shown in Section 6.9 the Self-Clocked algorithm has a problem with high and long–lasting packet loss. As said in Section 3.1.1 the height of packet loss is not a destructive characteristic. The graph of the evaluation of E-Sprout in Figure 30 shows that a drop of 60% of the data in one second can lead to direct depletion of the self-inflicted delay and correction of the OWD. Self-Clocked is not able to stop this packet dropping. A packet drop should always be the last opportunity of rate adaptation. If it is performed the encoder target bitrate should be at its minimum afterwards. In further work this misbehavior of Self-Clocked could be eliminated by simply adding this new condition.

6.13 High Start Bitrate without Knowledge of the Channel

Figure 31 illustrates a qualitative start behavior of Sprout in a very congested scenario. The available channel bitrate is so low that only a bitrate very near to the minimum video bitrate 0.15 Mbps can be transmitted without adding self-inflicted delay. Sprout sends an amount of data in the range of 1.5 Mbps and adds unnecessary delay to the transmission. The result is that the channel is directly congested. The depletion of this self-inflicted delay needs a very long time. By looking for the reason, it turned out that Sprout sends this data by default and without any knowledge of the channel. Running the other algorithms in the same simulation, it turned out that GCC, E-Sprout and Self-Clocked start with the minimum video bitrate to prevent self-inflicted delay at startup. These algorithms start to adapt to higher bitrate only after they have received a first feedback.

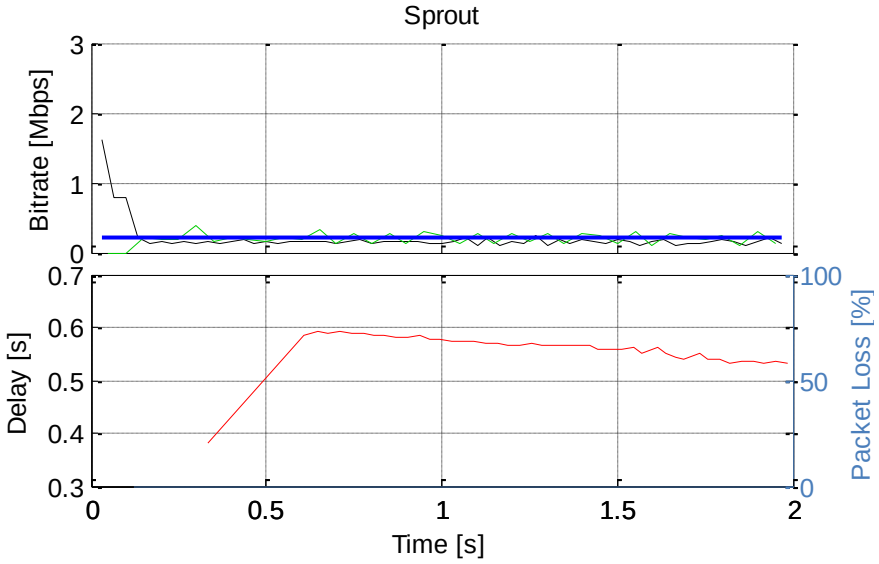

Figure 31 Sprouts starts behavior overshoots the channel. Channel (dark blue) is limited to 220 kbps. Throughput (green), encoder bitrate (black), packet loss rate (light blue) and OWD (red).

6.14 Additional OWD due to Packet Conservation

Self-Clocked, Sprout and E-Sprout prevent packets to wait in the network queue by queuing them up at the sender. The advantage of Self-Clocked and E-Sprout is that if congestion occurs in the channel, packets can be dropped at the sender. This is discussed in more detail in Section 6.12. In the case of E-Sprout and Sprout another opportunity is that packets can be send as little bursts to measure the channel, which is further discussed in Section 6.15. The bad effect of this behavior is that the time the packets are queued up before transmission is directly added to the OWD and downgrades the user experience. So a too long queuing should be avoided. The longest time should be the half of the RTT subtracted from the limit of the OWD to guarantee that this packet will arrive at the receiver in the limit of OWD, if one assumes equal up- and down-link delay times for mobile to mobile scenario. Otherwise this packets can be dropped because in conversational video this information cannot be used to show video in a real-time manner. In the case of false congestion detection this can lead to frequent loss of packets. So in the implementation of E-Sprout another instruction was introduced. It sends packets although congestion is detected. This is further described in Section 4.5.6. E-Sprout was evaluated with this feature in other simulations of this thesis. In Figure 32 the improvement of this ability is illustrated. In the average OWD plot the delay is decreasing. Consequently, there is a high increase of packet loss as seen in the packet loss rate graph. Even though the average OWD was reduced heavily, the worst OWD suffer by 90% of the users if 2% of the packets are neglected is still too high. As shown Section 6.9 in the Monte Carlo simulation the packet loss is not lasting for a long time. A more precise evaluation should discover the total time of packet loss. If this packet loss is only lasting for a view seconds of the call it is negligible.

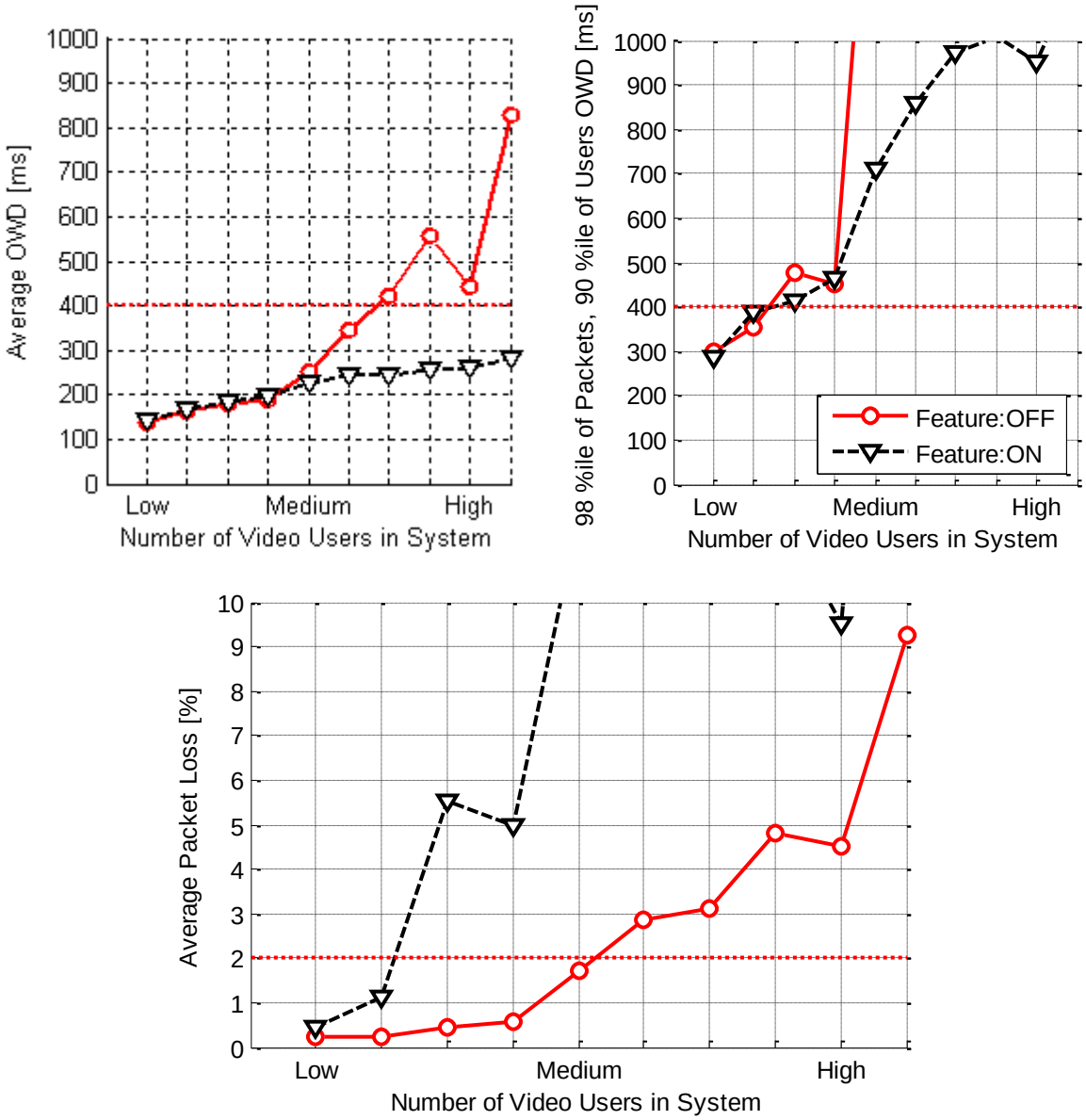

Figure 32 OWD and packet loss rate for E-Sprout with (On) and without (OFF) sending packets while congestion and packet dropping.

6.15 The Direct Measuring of the Channel

GCC and Self-Clocked probe the channel by constantly increasing the bitrate until the available bitrate of the channel is reached. Compared to them, E-Sprout can directly probe the channel without losing time, if there is enough data to send. This ability is taken from the original algorithm. E-Sprout sends such a high amount of data that the channel is directly utilized while preventing congestion. By measuring the incoming throughput E-Sprout estimates the channel and directly adjusts the video bitrate. This makes the algorithm much faster at the beginning of the call (see Figure 14) and after congestion (see in Figure 15). In our simulations E-Sprout reaches the available bitrate up to 20 times faster than GCC at start up. After congestion E-Sprout was able to reach the available bitrate 10 times faster than Self-Clocked. In the conversational video scenario a conclusion for Sprout cannot be given because it does not reach the available bitrate.

6.16 Low Reactivity of E-Sprout after very Low Available Bitrate

Figure 33 shows the recovery of E-Sprout after a very low available bitrate. In comparison with the graph in Figure 15 of E-Sprout the algorithm needs a much longer time to reach the available bitrate after second 30. As said in Section 4.4.2, the forecast of the received throughput and with that the sending rate is calculated by firstly filtering the number of received packets with Bayesian updating. Bayesian updating produces very low numerical numbers if the probability of this throughput is low. Through numerical limitation the probability of a throughput can even be zero and Bayesian updating can never make this throughput probable again although it is measured a view times. This can happen after a long time of a small throughput for E-Sprout. Then high bitrates are not probable anymore and through rounding errors the probability gets even smaller. The algorithm needs a long time to reactivate. The simulations shows, that a small time of low throughput can dramatically destroy the probability table of Sprout. Thus the algorithm can get stuck into fixed states after a while and never return into normal sending. The advice is to define a minimum probability so that numerical rounding issues don't play a role any more.

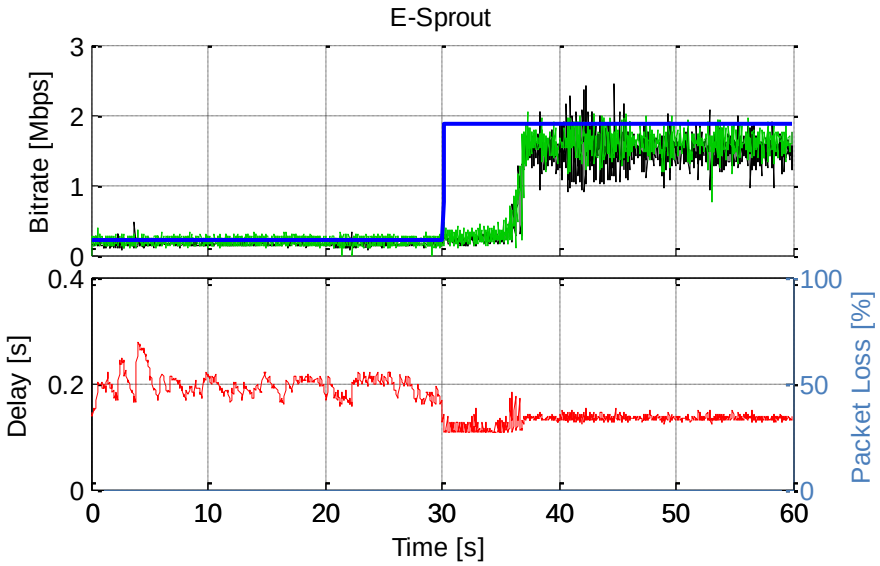

Figure 33 Example Graph E-Sprout reactivity after low available bitrate (dark blue), Throughput (green), encoder bitrate (black), packet loss rate (light blue) and OWD (red).

Chapter 7 Conclusions and Future Work

This thesis developed a concrete evaluation framework for media rate adaptation frameworks in mobile radio networks. A detailed mathematical construct was presented in section 3, by extracting key performance indicators (KPI) from the literature and merging them into a mathematical concept to fit together.

A very detailed LTE (Long Term Evolution) simulator was taken to compare the three media rate adaptation algorithms Sprout, Google Congestion Control (GCC) and Self-Clocked. Thus performance differences were shown and the influence between the algorithms was illustrated. Weaknesses of the algorithms were identified and solutions and ideas how to cope with them were given. For the Sprout algorithm a solution how to cope with conversational video in mobile radio networks is presented, called E-Sprout. A new way to figure out weaknesses of media rate adaptation algorithms with the help of the Monte Carlo simulation was developed.

In the following sub-sections the results will be discussed and finally future prospects will be presented.

7.1 Conversational Video in LTE Simulations

The comparison between Sprout in bulk transfer and Sprout with conversational video shows that traffic characteristics of real-time video play a crucial role in the design of an adaptation algorithm for video telephony. It is the nature of video traffic that the size of encoded video frames varies greatly. Also, it is only available periodically according to the video frame rate. If only throughput measurements are done, this will challenge the congestion detection and the estimation of the available bitrate.

7.2 KPI Discussion

In this subsection discusses the advantages of some of the used KPIs. The mathematic construct of the KPI shows the behavior of the algorithms. The most KPIs cannot give a statement by themself. Thus, the evaluation was done comparing the different statements of the KPIs. In this way, new findings were yield.

The aggressiveness KPI cannot be used to evaluate the influence between the different algorithms. Although E-Sprout has the highest aggressiveness (see Table 5), it will not suppress the other algorithms, if multiple user share the same bottleneck using different algorithms (see Figure 25). Since the ability not to be suppressed by the other algorithms is also connected to the stability of an algorithm, this leads to the assumption that the algorithm which is most aggressive and most stable will suppress the other algorithms. This statement could be investigated in further studies.

Since the stability of an algorithm is connected to the time an algorithm needs to adapt to the available bitrate, a definite value, which stability an algorithm should have, cannot be given. Although a very stable and conservative algorithm produces less packet losses and less

high one way delay (OWD) in a stable system, its performance in an unstable system is unacceptable and misses to follow the available bitrate. The other extreme, a very sensitive algorithm, which follows more the available bitrate, produces packet loss and a high OWD due to its instability, which can be seen in the example of E-Sprout. Also, the user experiences a high frequency of quality changes, which could decreases the user satisfaction.

The evaluation shows that, for conversational media rate adaptation algorithm, system stability is less connected to stability of an algorithm than to the utilization of the network. If many users use the network the overall use of the resources will be well distributed. Although an algorithm is very unstable in its behavior like Sprout or E-Sprout the system stability is less affected.

In the case of responsiveness the simulations shows that a definite statement cannot be made in a non-deterministic scenario. Since the frequency of congestion is influenced by the algorithm, the time between the congestions is not equal for all algorithms. Thus, for an algorithm with a high responsiveness, which also runs into frequent congestions inflicted by itself due to a high utilization of the network, the responsiveness KPI will be worse. In a deterministic scenario, where the frequency of congestion is fixed for each and every algorithm, future investigation could give clearer information.

The fairness index and the Ɛ-Fairness have to be compared with the fairly shared spectral efficiency (FSSE). Without this KPI the fairness index and the Ɛ-Fairness have the big disadvantage to give no information about the bitrate the minimum user can allocate. The minimum user is the user who suffers the most from the other users. For that, the FSSE gives a better conclusion for minimum users, but it cannot be taken alone for the evaluation, since it does not compare the users among each other. Fairness index and Ɛ-Fairness together with FSSE give a good conclusion of the fairness of the complete system.

7.3 Weaknesses and Comparison of the Algorithms

The evaluation of GCC, Self-Clocked and Sprout for real-time video over LTE has shown that all algorithms do not per se perform well in this case. The improved and presented E-Sprout algorithm is able to show that some of these weaknesses can be fixe. Nevertheless, the E-Sprout algorithm is yet too instable. None of these mentioned algorithms is ready to perform a satisfactory adaptation in the investigate scenario.

7.3.1 Google Congestion Control Algorithm

In Section 6.1 GCC shows misbehavior in the OWD limit for more than 10% of the users although it is not fully utilizing the network. Section 6.3.2 presents that a sudden congestion leads to an unsolvable congestion and a long time of high OWD in GCC sessions. This problem can be solved by dropping packets before sending, if congestion is detected like it is shown by E-Sprout and Self-Clocked.

Section 6.5 shows that GCC is very conservative and does not vary the encoder bitrate very much. The reason for that is discussed in Section 6.3.2. The Section concludes with the intention that the receiver side controller does not do a good job in the adaptation when

congestion occurs. The reduction in one step is limited to 5% which is much too low for mobile radio. This thesis suggests making the value adaptive to the change of the output of the Kalman filter. The bad responsiveness is also shown in Section 6.9. One can see that a fall of the available bitrate from 2 Mbps to 0.9 Mbps cannot be compensated by the GCC algorithm. So, it reacts too slowly on congestion no matter how long the change takes.

In Section 6.3 the long start up time for GCC is discussed. To compensate this problem, the algorithm should have a startup state. In this state it could react stronger to available bitrate, like it is done by Self-Clocked. Another opportunity is also discussed. The increase factor η is adaptive to the RTT. Since this factor is not varying a lot a better metric for adaptation could be the measured channel bitrate like it is done by Sprout.

In Section 6.5 the smoothness of the algorithm is presented, which is an advantage of the algorithm. In its steady state and in a stable system the algorithm is doing a good job. However this is not the scenario an adaptation algorithm is designed. Section 6.8 shows that the algorithm is easily supplanted by other algorithms like Self-Clocked or TCP Flows.

7.3.2 Self-Clocked

In Section 6.12.1 the unnecessary packet drops of Self-Clocked are discussed. The drops can be seen in a qualitative graph in Figure 30. If there is a continual packet loss, the decoder will not be able to show the video any more. Therefore, it is recommended to lower the bitrate to the minimum, if the algorithm starts to drop packets.

The reaction of Self-Clocked to the available bitrate is fast. An algorithm, which adapts to the available bitrate in such a short time like E-Sprout, is also more unstable. So, the evaluation how fast an algorithm should adapt has to be done in further studies. This factor has different point of interests. On the one hand, as said in Section 3.2.2, an adaptation algorithm should follow the available bitrate. On the other hand it should neglect changes of the available bitrate which disappear in the connections round trip time (RTT). This statement can be further classified. When the available bitrate goes down, the adaptation algorithm should always follow it to avoid too high OWD and packet loss. However, it should not react to rises of the available bitrate, which are only consistence for a short time, since the user should not be annoyed by a frequently changing quality.

For Self-Clocked the startup state exists. This helps the algorithm to provide a good quality in a short startup time at the beginning of the video. The stability and also the smoothness of the algorithm are comparable to GCCs (see Section 6.5). In a realistic scenario Self-Clocked is the only algorithm which can compete with TCP sessions and other adaptation algorithms (see Section 6.8). This ability is an outstanding feature and all the other algorithms should have the target to deliver the same results. Since the algorithm is TCP fair it is acceptable to be that stable and aggressive.

7.3.3 Sprout

The Sprout algorithm shows a high instability in the evaluation (see Section 6.5). Since the throughput is highly varying due to the video signal, a media rate adaptation framework only based on throughput measurement is rather difficult. The algorithm in its original version is

unsuitable for conversational video. However, this thesis presents enhancements which make an adaptation framework based on the original Sprout possible.

7.3.4 E-Sprout

E-Sprout fixed many weaknesses of the Sprout algorithm, which are present in the conversational video scenario. However, E-Sprout is not yet an option for an adaptation framework. Its instability and aggressiveness are yet too high. It does not show good behavior in the LTE best effort scenario (see Section 6.1). The algorithm probes the channel and runs into congestion too frequently (see Section 6.4). Even though it behaves that unstable, the Monte Carlo simulations in Section 6.9 show that it can react to channel changes very fast and reliable. If the algorithm was more stable, the delay and the packet drop rate would be acceptable for E-Sprout. However, the amount of packet loss and OWD could also get worse if the algorithm gets more stable.

The utilization of the network from E-Sprout is acceptable and it is reaching the available bitrate, although the measurement of the throughput is disturbed. The algorithm filters the throughput measurement with a max order filter when the RTT is below 100 ms. Thus, the time, when the channel is not fully utilized, is not as destructive for E-Sprout as it is for Sprout. However, this is only a hard fix. In further studies the factor 100 ms has to be adaptive since there are connections, which can have long RTT although a high video bitrate could be possible.

The algorithm shows a few interesting facts. Its fast adaptation through direct measurement reinforces the idea that congestion control algorithms have to probe the available bitrate by constantly increasing the throughput. Since a burst of packets can give a bunch of information on the channel, it is preferable to use this kind of measurement. In this way a packet scheduler like it is used by Self-Clocked and E-Sprout is needed. The forecast guarantees that the amount of packets transmitted for the bursty measurement is not producing collapse of the packets and as a consequence packet loss.

The motivation to develop E-Sprout was to show that the concept of Sprout to use throughput as a measurement works, if RTT is added as a measurement. The results show that it works but further improvements could to be done.

As said in Section 6.16 the problem of numerical calculation also has to be solved. It is suggested to define a minimum probability so that numerical rounding issues do not play a role anymore.

7.4 Monte Carlo simulation

In Section 5.2.1 the setup for the Monte Carlo simulation is presented. In Section 6.9 the results show that Monte Carlo simulation combined with bitpipe scenario can give a conclusion about the behavior of the algorithm. The packet loss problems of Self-Clocked and the inertia of GCC were discovered in the Monte Carlo simulation. This leads to the assumption that Monte Carlos simulation can be useful to evaluate adaptation algorithms. The drawback of these simulations is that they are not close to reality. An algorithm, which is evaluated by these Monte Carlo simulations and shows acceptable performance, does not

have to work in a realistic scenario. So, the Monte Carlo simulations always have to be compared to a detailed and realistic simulator. Nevertheless, the Monte Carlo simulations can be taken to find out reasons for the weaknesses and to identify the limits of an algorithm, e.g. how fast the adaptation is and which amount of changes in the available bitrate can be compensated.

7.5 Future Work

Algorithms

The improvements suggested for each algorithm are the basis to future work in this field. The misbehavior of Self-Clocked concerning packet dropping is yet under repair. The instability of E-Sprouts has to be evaluated further with an inner process evaluation. Since GCC is not only meant for mobile radio applications, the evolving of this algorithm is a suggestion but not mandatory.

The question, why the other algorithms cannot survive in a competing scenario against Self-clocked is not yet answered. A further evaluation has to be made. The ability to be not suppressed by other algorithms in a competing scenario seems to be effected by a certain combination of the stability and aggressiveness. Thus, the suggestion is to toggle the inner algorithm features of Self-Clocked, to see which feature is responsible for its stability and aggressiveness.

Scenario

Also more scenarios like up-link and combined up-link and down-link simulations could be considered in future work. Small buffers or active queue management (AQM) could show if the behavior of GCC is better with packet loss as a channel metric.

The presented adaptation framework is only implemented in the clients, which have no knowledge about the other users in the cell. Using additional control information from the network and base station could bring further improvements. Explicit congestion notification (ECN) is such a notification. A future study could show if ECN leads to more fairness and a more stable adaptation.

A more advanced scenario would include long-lived TCP session to get a more realistic evaluation like it is suggested in [3]. Also the absence of audio on the same bottleneck does not meet the reality and could be added in further studies.

Evaluation

In this evaluation only the coefficient of variation (CoV) of the throughput is used to evaluate the constancy of the video quality. Although this gives information about the constancy of the video quality a further evaluation could distinguish between small changes and big drops of video quality. Furthermore, the packet loss was only evaluated over the whole call time. A more precise evaluation would measure the loss distance and loss period like it is suggested in [35]. The burst density, which is a fraction of packets in bursts, the gap density, which is the fraction of packets in the gaps between bursts, the burst duration, which is the mean

duration of bursts in seconds, and the gap duration, which is the mean duration of gaps in seconds (see [36]) could give information about the structure of the transmitted data.

Another interesting field of evaluation is to use another codec for the video. In the example of Sprout it was seen that adaptation algorithm are strongly influenced by the media which they send. Since H265/HEVC Video has a different characteristic as H264/AVC for the same resolution of video, the results of this thesis could look different if using H265. This could be the subject of further investigations.

The user experience and the video quality where only set by assumptions. A Modell for the encoder and decoder with assumptions on error concealment and quality is taken. Although the most of these assumptions meet common acceptance, the reality could be different. The implementation of encoders and decoders varies which makes the selection of a general encoder even more problematic. To find metrics to measure real video quality in an adaptation scenario is a very interesting future task and could lead to a much more subjective evaluation. Also the user experience is a very interesting field. For future work, it is suggested to evaluate the user experience in an emulated network, which can be used in real-time. Then, the user experience could be directly observed by test participants.

References

[1] "IMS Profile for Conversational Video Service," in *GSMA IR.94*.

[2] K. Winstein, A. Sivaraman and H. Balakrishnan, "Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks," in *10th USENIX Symposium on Networked System Design and Implementation, NSDI'13*, Lombard, 2013.

[3] V. Singh and J. Ott, "Evaluating Congestion Control for Interactive Real-time Media," 26 February 2013. [Online]. Available: http://tools.ietf.org/html/draft-singh-rmcat-cc-eval-04. [Accessed October 2013].

[4] H. Lundin, S. Holmer and H. Alvestrand, "A Google Congestion Control Algorithm for Real-Time Communication - draft-alvestrand-rmcat-congestion-00," 18 February 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-alvestrand-rmcat-congestion/. [Accessed 15 July 2013].

[5] S. Floyd, "Metrics for the Evaluation of Congestion Control Mechanisms RFC 5166," March 2008. [Online]. Available: http://www.rfc-editor.org/rfc/pdfrfc/rfc5166.txt.pdf. [Accessed August 2013].

[6] R. Jesup, "Congestion Control Requirements For RMCAT," in *Network Working Group*, 2013.

[7] International Telecommunications Union, "One-way transmission time," in *ITU-T Recommendation G.114*, May 2003.

[8] J. Jansen, P. Cesar, D. Bulterman, T. Stevens, I. Kegel and J. Issing, "Enabling Composition-Based Video-Conferencing for the Home," *Multimedia, IEEE Transactions,* vol. 13, no. 5, pp. 869-881, 2011.

[9] 3rd Generation Partnership Project, "Media handling and interaction," in *Technical Specification Group Services and System Aspects*, 2013.

[10] Y. Xu, C. Yu, J. Li and Y. Liu, "Video Telephony for End-consumers: Measurement Study of Google+, iChat, and Skype," in *ACM conference on Internet measurement*, Boston, Massachusetts, USA, 2012.

[11] D. X. Wei, P. Cao and H. Low, "Time for a TCP Benchmark Suite?," Caltech CS; Stanford EAS, 2005.

[12] S. Floyd, M. Handley and J. Padhye, "A Comparison of Equation-Based and AIMD Congestion Control," 12 May 2000. [Online]. Available: http://www.icir.org/tfrc/aimd.pdf. [Accessed August 2013].

[13] C. Jin, D. Wei and S. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *INFOCOM 2004*, 2004.

[14] Y. R. Yang, S. K. Min and S. Lam, "Transient behaviors of TCP-friendly congestion control protocols," in *INFOCOM 2001*, 2001.

[15] I. Matta and V. Tsaoussidis, "Open Issues on TCP for Mobile Computing," *JOURNAL*

OF WIRELESS COMMUNICATIONS AND MOBILE COMPUTING, vol. 2, pp. 3-20, 2002.

[16] M. Ericsson, "Dynamic single frequency networks," *Selected Areas in Communications, IEEE Journal on,* vol. 19, pp. 1905-1914, 2001.

[17] S. Ha, Y. Kim, L. Le and I. Rhee, "A Step toward Realistic Performance Evaluation of High-Speed TCP Variants," in *Elsevier Computer Networks (COMNET) Journal, Special issue on*, 2006.

[18] K. Baharath-Kumar and J. M. Jaffe, "A new approach to performance-oriented flow control," *Communications, IEEE Transactions on,* vol. 29, pp. 427-435, 1981.

[19] R. K. Jain, D.-M. W. Chiu and W. R. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation," ACM Transaction on Computer Systems, Hudson, 1984.

[20] J.-Y. Boudec, *Rate adaptation, Congestion Control and Fairness: A Tutorial,* 2000.

[21] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," in *IEEE/ACM Transactions on Networking*, August 1999.

[22] D. Xie, S. Mehrotra, J. Li and Y. C. Hu, "URCP: Universal Rate Control Protocol for Real-Time Communication Applications," 2013.

[23] G. Ascheid, Systemtheorie 2, Aachen: Druck und Verlaghaus Mainz GmbH Aachen, 8. Auflage 2009.

[24] L. De Cicco, G. Carlucci and S. Mascolo, "Experimental Investigation of the Google Congestion Control for Real-Time Flows," in *ACM SIGCOMM 2013 Workshop on Future Human-Centric Multimedia Networking*, Hong Kong, China, 2013.

[25] S. Floyd, M. Handley, J. Padhye and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," September 2008. [Online]. Available: http://tools.ietf.org/html/rfc5348. [Accessed 14 October 2013].

[26] S. Shalunov, G. Hazel, J. Iyengar and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)," 31 October 2011. [Online]. Available: http://tools.ietf.org/pdf/draft-ietf-ledbat-congestion-09.pdf. [Accessed 8 10 2013].

[27] S.-H. Choi and M. Handley, "Designing TCP-Friendly Window-based Congestion Control for Real-time Multimedia Applications," in *International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT)*, Tokyo, May 2009.

[28] B. Ford, "Structured Streams: a New Transport Abstraction," in *ACM SIGCOMM 2007 Data Communications Festival*, Kyoto, August 207.

[29] A. Erbad, M. T. Najaran and C. Krasic, "Paceline: latency management through adaptive output," in *ACM SIGMM conference on Multimedia systems*, Phoenix, Arizona, USA, 2010.

[30] K. Winstein, A. Sivaraman and B. Hari, "Congestion Control for Interactive Real-Time Flows on Today's Internet," in *IAB / IRTF Workshop on Congestion Control for Interactive Real-Time Communication*, July 2012.

[31] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Queue,* vol. 9, pp. 40:40--40:54, 2011.

[32] H. Ekström, "QoS Control in the 3GPP Evolved Packet System," *IEEE Communications Magazine,* vol. Release 8, pp. 76-83, 2009.

[33] 3GPP, "Physical layer aspect for evolved Universal Terrestrial Radio Access (UTRA),"

Tech. Rep. 3GPP TR 25.814 V7.1.0, vol. Release 7, 2006.

[34] H. Schulzrinne, "RTP: A Practical Approach to Energy-Aware Cellular Data Scheduling," in *ACM MOBICOM*, 2010.

[35] R. Koodli and R. Ravikanth, "One-way Loss Pattern Sample Metrics RFC 3357," August 2002. [Online]. Available: http://www.rfc-editor.org/rfc/pdfrfc/rfc3357.txt.pdf. [Accessed August 2013].

[36] T. Frieman, R. Caceres and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR) RFC 3611," November 2003. [Online]. Available: http://www.rfc-editor.org/rfc/pdfrfc/rfc3611.txt.pdf. [Accessed August 2013].

List of Figures

List of Tables

Mathematical symbols

Variable	Definition	Dimension
A	Available bitrate of the channel	$\left[\frac{Bit}{s}\right]$
B	Radio spectrum bandwidth	$\left[\frac{1}{Hz}\right]$
$\hat{x}$	Estimated variable	
i	Evaluation step	
N	Number of sessions	
R	Incoming bitrate	$\left[\frac{Bit}{s}\right]$
x	Throughput	$\left[\frac{Bit}{s}\right]$
L	Packet loss rate	$\left[\frac{Bit}{s}\right]$
OWD	One way delay	$[s]$
FLR	Frame loss ratio	
VB	Actual achieved video bitrate/encoder bitrate	$\left[\frac{Bit}{s}\right]$
CoV	Coefficient of variation	
σ	Variation	

R_a	Responsiveness	
$Aggr$	Aggressiveness	$\left[\frac{Bit}{s}\right]$
Od	Overhead	
η	Spectral efficiency	$\left[\frac{Bit}{s * Hz}\right]$
F	Fairness index	
$FSSE$	Fairly shared spectrum efficiency	