Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Sami Karvonen

Evaluation of video stream adaptation algorithms in mobile cloud gaming

Master's Thesis
Espoo, September 21, 2016

Supervisors: Professor Antti Ylä-Jääski, Aalto University
Advisor: Matti Siekkinen D.Sc. (Tech.)
Teemu Kämäräinen M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Sami Karvonen

Title:
Evaluation of video stream adaptation algorithms in mobile cloud gaming

Date: September 21, 2016 **Pages:** 48

Major: Data Communication Software **Code:** T-110

Supervisors: Professor Antti Ylä-jääski

Advisor: Matti Siekkinen D.Sc. (Tech.)
Teemu Kämäräinen M.Sc. (Tech.)

Mobile cloud gaming has recently gained popularity as a result of improvements in the quality of internet connections and mobile networks. Under stable conditions, current LTE networks can provide a suitable platform for the demanding requirements of mobile cloud gaming. However, since the quality of mobile network connections constantly change, the network may be unable to always provide the best possible service to all clients. Thus, the ability to adapt is necessary for a mobile cloud gaming platform in order to compensate for changing bandwidth conditions in mobile networks. One approach for doing this is to change the quality of the video stream to match the available bandwidth of the network.

This thesis evaluates an adaptive streaming method implemented on a mobile cloud gaming platform called GamingAnywhere and provides an alternative approach for estimating the available bandwidth by measuring the signal strength values of a mobile device. Experimentation was conducted in a real LTE network to determine the best approach in reconfiguring the encoder of the video stream to match the bandwidth of the network. The results show that increasing the constant-rate-factor parameter of the video encoder by 12 reduces the necessary bandwidth to about half. Thus, changing this video encoder parameter provides an effective means to compensate for significant changes in the bandwidth. However, high values of the constant-rate-factor parameter can considerably reduce the quality of the video stream. Thus, the frame rate of the video should be lowered if the constant-rate-factor already has a high value.

Keywords: mobile, cloud, gaming, adaptive, video, streaming, signal, strength

Language: English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Sami Karvonen

Työn nimi:
Mukautuvien videon toisto algoritmien evaluointi mobiilipilivipelaamisessa

Päiväys:	21. Syyskuuta 2016	**Sivumäärä:**	48
Pääaine:	Tietoliikenneohjelmistot	**Koodi:**	T-110
Valvojat:	Professori Antti Ylä-Jääski		
Ohjaaja:	Matti Siekkinen D.Sc. (Tech.) Teemu Kämäräinen M.Sc. (Tech.)		

Mobiilipilvipelaaminen on viimeaikoina kerännyt suosiota parantuneiden internet yhteyksien ja mobiiliverkkojen ansioista. Normaali olosuhteissa nykyiset LTE verkot tarjoavat sopivan alustan mobiili pilvipelaamisen koviin vaatimuksiin. Mobiiliverkkojen yhteyden laatu kuitenkin vaihtelee jatkuvasti ja kaikille käyttäjille ei voida aina tarjota parasta mahdollista yhteyttä. Mukautuminen vaihtelevaan yhteyden laatuun on siis tarpeellista pilvipelaamisalustalle. Tämän voi tehdä esimerkiksi muuttamalla videon kuvanlaatua sopivaksi käytössä olevaan kaistaan.

Tässä työssä arvioidaan GamingAnywhere alustalle toteutettu mukautuva videon toistomenetelmä ja esitellään vaihtoehtoinen tapa arvioida käytettävissä olevaa kaistaa mittaamalla mobiilisignaalin vahvuutta mobiililaitteessa. Aidossa LTE verkossa suoritettujen kokeiden avulla selvitettiin paras tapa konfiguroida video enkooderi mukautumaan käytettävissä olevaan kaistan määrään. Tuloksista selviää, että constant-rate-factor-parametrin arvon nostaminen kahdellatoista laskee tarvittavan kaistan määrän noin puoleen. Se on siis tehokkain tapa mukautua merkittäviin muutoksiin kaistanleveudessä. Liian suuret constant-rate-factor-parametrin arvot kuitenkin heikentävät kuvanlaatua muerkittävästi, joten kuvataajuutta voi myös alentaa jos parametrin arvo on jo liian suuri.

Asiasanat: mobiili, pilvipelaaminen, mukautuva striimaus, signaalin vahvuus

Kieli: Englanti

Acknowledgements

I would like to express my gratitude to my supervisor Antti Ylä-Jääski and my advisors Matti Siekkinen and Teemu Kämäräinen for an interesting topic to work on and their invaluable guidance while writing this thesis.

I would also like to thank to my family for their support during the whole duration of my studies.

Espoo, September 21, 2016

Sami Karvonen

Abbreviations and Acronyms

MCG	Mobile cloud gaming
MCC	Mobile cloud computing
GA	GamingAnywhere
QoS	Quality of Service
RTSP	Real Time Streaming Protocol
RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
API	Application Programming Interface
LTE	Long-Term Evolution cellular network
HSPA	High Speed Packet Access
WCDMA	Wide-band Code Division Multiple Access
RTT	Round-trip Time
MOS	Mean Opinion Score
GPU	graphics processing unit
QSV	quick sync video
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RSSI	Received Signal Strength Indicator
CRF	Constant Rate Factor
QP	Quantization Parameter

Contents

Chapter 1

Introduction

In recent years, mobile cloud computing (MCC) has increasingly become an important tool for storing data and computation, since it allows mobile devices to offload various tasks to more powerful computers at data centers. Although offloading tasks can save the limited resources of mobile devices, deciding which tasks to offload is not trivial because it introduces new costs in the form of data transmissions and delays. Transmitting data to cloud servers requires power and may end up draining more power than that needed for executing the task locally, if the decision of which tasks to offload is not made carefully. Moreover, communicating over a network introduces a delay which may have a negative effect on user experience. Recently, mobile offloading has been the focus of much research, leading to considerable progress in extending the battery life and limited storage capacities of mobile devices. One example of using offloading to extend the capabilities of a mobile device is Mobile cloud gaming (MCG).

MCG is a form of mobile offloading in which the game is executed and rendered on the cloud and then streamed to a mobile device, which acts as a thin client. Thus, the mobile device shows the video stream to the user and sends the commands executed by the user to the cloud. However, the communication between the user and the cloud is affected by a delay caused by the underlying network. For many games, it is crucially important for the user experience that this delay is kept as short as possible. Acceptable delay depends on the type of the game, though it typically ranges between 80 and 200ms[16]. In a real world scenario where the properties of the mobile network are constantly changing, it is difficult to keep the delay at the acceptable range, though some solutions have been proposed[28]. Moreover, another important property of the mobile network, bandwidth, has received little attention. The bandwidth of the network does not directly affect delay, but using too much of the available bandwidth will clog the network, resulting

in very high delays. Since streaming video over the network can use considerable bandwidth, it is important to be able to control the bandwidth usage. Controlling the bandwidth usage can be done by adapting the quality of the video stream in MCG systems [31]. For best possible user experience, the quality of the video stream should be as high as possible but still kept within the limitations of the network to keep the delay within the acceptable range. This control is not trivial. While several studies have tested implementation of MCG on simulated networks [11, 31], few studies have attempted to assess the MCG scenario in a real mobile network environment.

1.1 Problem statement

Adaptive video streaming is necessary for MCG system to work on a changing environment, such as the mobile network. While some research on the subject has been done, most of the proposed solutions have been tested in only a simulated environment on the physical network. Testing of a mobile system on the physical network might not produce accurate results, since the physical and mobile networks differ in various ways, and simulated testbeds might not produce the same results as those using tests performed in a real mobile environment. One example of these differences is that in a mobile network packets could be delivered out of order[14]. Adaptive streaming solutions, such as those relying on the time difference between packets[11], might not work properly on a real mobile network, even though it produces positive results on a simulated environment with latency and bandwidth similar to that of a mobile network. Thus, it is necessary to test these adaptive streaming solutions on a real mobile network.

1.2 Objectives and focus

Although the adaptive streaming implementation introduced in [11] was developed for the GamingAnywhere[12] platform (an MCG system), it was tested on a simulated network using DummyNet. As explained earlier, this might present a problem because DummyNet might not correctly simulate all the attributes of a real mobile network. Therefore, the objective of this thesis is to create and evaluate the performance of a testbed on a real mobile network. Based on the results of the evaluation, modifications will be proposed to improve the performance of the implementation on real mobile networks.

This thesis will focus on the implementation proposed in [11] because it is

open source and readily available. While other adaptive streaming solutions might exist, they are either closed systems or created for other purposes than online gaming. Experiments will be conducted on a single Long-Term Evolution (LTE) base station using varying signal strengths, thus avoiding mobile handoffs during the experiments.

1.3 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 introduces the concept of MCG and its most important building blocks. Chapters 3 and 4 present the background and previous research on LTE networks as well as adaptive streaming. Chapter 5 describes the experiments in detail. Chapter 6 presents the results from the experiments which are further discussed in Chapter 7.

Chapter 2

Mobile Cloud Gaming

This chapter introduces the concept of mobile cloud gaming and the main technologies necessary for it. Examples of existing mobile cloud gaming systems, such as Gaikai and G-cluster, are introduced and previous research is reviewed. GamingAnywhere software is described in more detail than the others since it will be used in the experiments of this thesis.

2.1 Mobile cloud gaming systems

In recent years, the popularity of mobile games has increased significantly [4]. Some proprietary applications, such as G-cluster [7], already offer some Mobile Cloud Gaming (MCG) services on mobile phones and tablets. The concept of MCG, illustrated in Figure 2.1, means that the game engine, or parts of it, is deployed in the cloud servers instead of the mobile device. Deploying the whole engine on the cloud server means that client does not have to do any rendering so it can just display the video stream to the user. This enables games with higher hardware requirements to be played on mobile devices because the most of the heavy computation is transferred to to cloud servers. However MCG technology is still quite new. It has many difficulties, such as delay between the client and the server, thus guaranteeing Quality of Service(QoS) is difficult.[5]

The basic idea of MCG system is a high-end server located at a cloud data center sends a video stream over the Internet to a simplistic client. The client serves as a simple playback and input device. This model has many advantages such as potentially saving power on mobile devices, reducing the hardware expenses of clients and reducing the need to adapt games for multiple platforms. However, developing a MCG system also brings some challenges:

- **Latency** The wireless and cellular networks may introduce high delays which might effect the gaming experience.
- **Energy** Even though it saves energy to execute the heavy computation at cloud servers, it may introduce energy drains in form of data transfers.
- **Cost** Transferring significant amounts of data at a cellular network may introduce costs for the user.

Latency is most impactful of these three from the Quality of Experience(QoE) point of view. Since the range where the latency becomes intolerable for many games is somewhere between 100 and 150ms this may very well be a problem for mobile networks.[19]

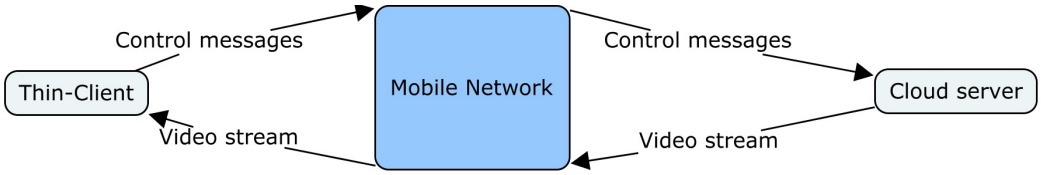


Figure 2.1: The basic concept of MCG.

GamingAnywhere(GA) is an open source MCG platform that has been designed to be highly extensible and modifiable. Thanks to its openness and modifiability it is ideal for research purposes and testing new MCG platforms and ideas. GA adopts a modular design which makes it easy to change or modify some parts parts of the software and keep the others intact. This enables easy testing of new ideas and solutions.[13] For these reasons GA is used as a platform for the experimental part of this thesis.

2.2 GamingAnywhere architecture

GamingAnywhere is the only open source MCG system thus it will be used as an example of MCG architecture. Figure 2.2 demonstrates the modular architecture of GamingAnywhere. Basically it consists of a game server and a game client which communicate with each other using two different network flows control flow and data flow. The control flow is used to send the user's actions from the client to the server and the data flow streams the audio and video frames to the client. The client and the server consist of multiple modules. The client includes decoders for audio and video, real time streaming protocol(RTSP)[10] module and encoder for input events. The server

includes encoders for audio and video, RTSP module and decoder for input events. This thesis focuses on adaptive videos streaming, thus the rest of this section focuses on the video and RTSP modules. [12]

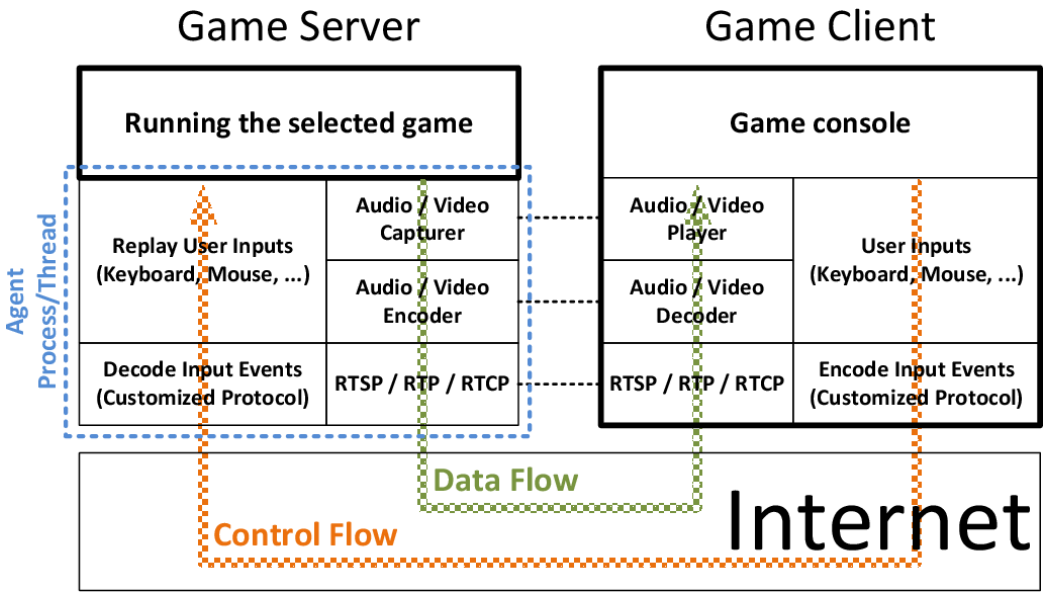


Figure 2.2: Modular architecture of GamingAnywhere.[12]

The RTSP module is the first module launched on the server side and it is used to accepts RTSP commands from the client, launch encoders and setup the data flow. The data flows can be transmitted over TCP or UDP in form of RTP/RTCP packets. On the server side GA has a custom implementation for handling the RTSP commands and delivering the binary data and *libavformat* library is used for packetization of RTP and RTCP packets. On the client side *live555* library is used for RTSP protocol.[12]

The video capture module on the server side can work in two different ways. First one is the desktop capturing mode which takes screen shots of the desktop at a given frame rate. Second is the application programming interface(API) intercept mode which works in an event-driven manner and captures each frame when the game has completed its rendering. After capture these frames are encoded by one or more encoders and sent to the client using the RTSP protocol. The frames are encoded using *libavcodec* library which can support multiple different codecs. Different codecs are discussed in more detail in Section 4.2. At the client side the frames are decoded and shown to the user. Also, the client uses zero-buffering strategy, meaning only the latest frame is kept in the buffer, to reduce input latency.[12]

2.3 Measuring MCG systems

MCG systems have multiple complicated parts which all affect performance of the whole system. The most important metric to measure is the response delay because if it is too high many games become unplayable. However there are other important metrics, such as the video quality and bandwidth usage, which affect the QoE of the user. The focus of this thesis is measuring the performance of adaptive video streaming which basically means lowering the video quality to keep the response delay at the a reasonable level. Thus, it is necessary to make an acceptable compromise between video quality and response delay to keep the game playable.

Response delay in MCG systems means the time between the user making an action and the time the result of that action is shown on the screen. By itself the response delay metric is very useful at indicating how responsive the system as a whole is but there is no single part in the system that is solely responsible for the delay alone so it is usually split into three parts:[6]

- **Network delay** The time it takes for a network packet to go from the client to server and back.
- **Processing delay** The time it takes for the server to generate the next frame after it receives a command from the user.
- **Playout delay** The time difference between client receiving a frame and the frame being visible on the screen.

It is important to know how much of the response delay is caused by each of the parts because the information can be used to make the QoE better for the user. Network delay in mobile network usually something that cannot be affected by the MCG system. However, it can be measured and the information can be used to guide an adaptive streaming protocol to provide the best possible video quality. For example, if the network delay is 40ms and we know that a first person shooter game is playable up to 100ms, then we can deduce that there is plenty of time for frame processing thus we can choose a high quality configuration for the video encoder. On the other hand if the network delay is 90ms a lower quality configuration is needed to meet the delay requirements.

Another metric to measure MCG systems by is video quality which is very subjective measurement. To measure video quality objectively, a quantifiable metric is needed. For this purpose the bitrate of the video stream can be used. Bitrate doesn't always fully reflect video quality because there are other aspects of video, such as frame rate and the encoder used, which may

have an affect[30]. However, bitrate does directly correlate to the amount of data transferred, thus theoretically higher bitrate means that a higher video quality can be achieved with it. In MCG systems which use adaptive streaming to achieve the highest bitrate possible, the available bandwidth has to be estimated as accurately as possible. Bandwidth estimation is discussed in more detail in Section 3.4.

The overall quality of gaming experience on a MCG system is affected by many factors and is very subjective. Thus, it cannot be measured by metrics such as bitrate and response delay alone. One way to make such measurements is an extensive user study which requires a lot of time and money to conduct.[11] Thus, on the experimental part of this thesis in Chapter 5, these metrics will be used as quality measurements because it is out of the scope of this thesis to conduct such user studies.

Chapter 3

LTE networks

This chapter describes the characteristics of Long-Term Evolution(LTE) networks and explains the important differences between simulated mobile network and a real one.

3.1 LTE network characteristics

LTE network is the latest cellular network technology specified by the 3rd Generation Partnership Project(3GPP). It is an evolution from its predecessor 3G standards Wide-band Code Division Multiple Access(WCDMA) and High Speed Packet Access(HSPA). LTE network has been shown to have low latencies and high bandwidth in field trials. An average Round-trip Time(RTT) of 35 and and average throughput of over 40Mbps on downlink and about 20Mbps on uplink[32]. These values are well within the limits of of even the most demanding games, thus implementing mobile cloud gaming platforms on LTE networks should be possible. Although, even if the average values are acceptable, real-time based games usually require the connection to remain usable for the whole duration of the gaming session. Also, there are other attributes such as jitter and packet loss which may effect the QoE of the user.

3.2 Packet loss and QoE

Packet loss can be problematic in LTE networks and even a single packet loss can cause severe performance issues[15]. Although packet loss rates can be as low as 0,06% in LTE networks which is comparable to a physical network[15], situations exists where it can be significantly higher, i.e. when the user is moving very fast [24]. From MCG point of view high packet

loss forces high amount retransmissions which increases the response delay of the whole platform. This makes packet loss almost as important metric as response delay[17]. Figure 3.1 shows the effect of packet loss to Mean Opinion Score(MOS) value.

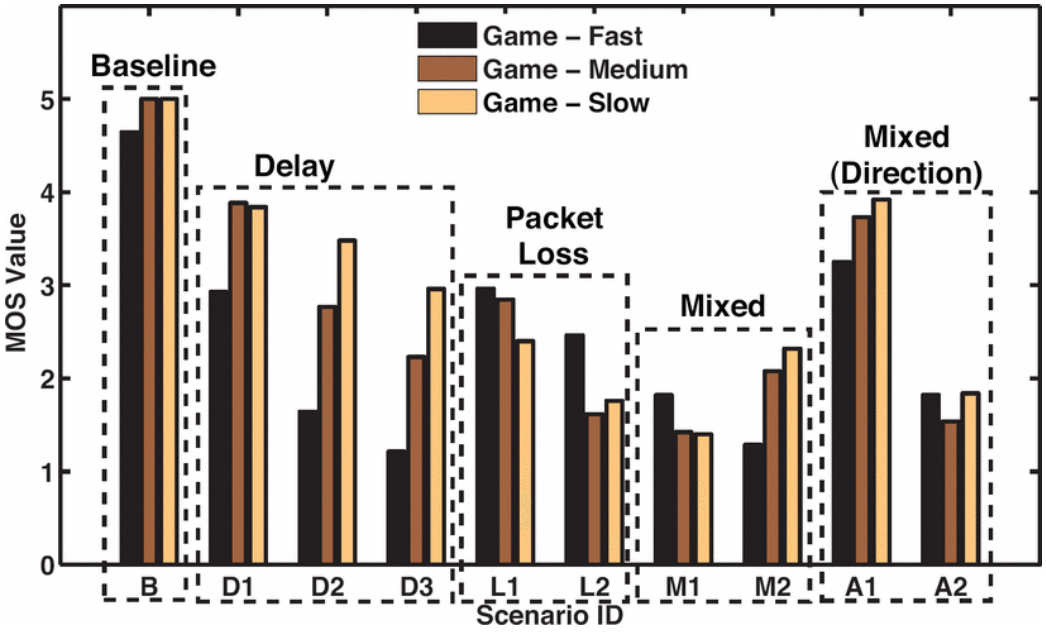


Figure 3.1: Packet losses effect to MOS value.[17]

In MCG scenario packet loss can happen in two different ways. Packet can either be lost when the client is sending a command to the server or when server is sending the video stream to the client. Packet loss rate of 1% while sending packets from the client to the server does not have much effect on the QoE of the user. However, having the same amount of packet loss when sending data to the opposite direction causes massive video distortions and effects negatively on the QoE. Thus, it is important to know where the packet loss happens and it is beneficial to have an asymmetric connection with lower packet loss on the downlink side if possible. [17]

3.3 Signal strength

Signal strength of mobile device can vary significantly depending on many factors, i.e. distance to the nearest base station and different objects blocking the path to the base station can weaken the signal strength. When the radio in a mobile device encounters a weak signal strength it tries to keep the error

rate low by switching to a lower rate of modulation [26]. Thus, decreasing the data rate of the connection. If the signal strength drops rapidly, for example the user goes from outside to inside, there's a rapid decrease in the data rate of the connection. An adaptive streaming algorithm which relies on the packet loss rate of the connection to adapt the quality of the video stream might not keep up, resulting in a momentary increase in latency.

In MCG scenario it is important to keep the latency low at all times, thus implementing a system which reacts faster to decreasing quality of the connection would be beneficial. The weakening signal strength is the first indication of a drop in connection quality, thus it should be used for choosing the time to lower the video quality in an adaptive streaming algorithm. However, the signal strength can be only measured by the client which means this data needs to be actively sent to the server by the client for this scenario to work. For example an energy aware cellular data scheduling system called Bartendr, proposed in [26], predicts the changes in signal strength and uses the data to save energy. A similar system could be implemented for adaptive streaming to predict when the stream quality needs to be changed.

LTE base stations provide many different signal strength indicators to the mobile device. Most important of them are Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ) and Received Signal Strength Indicator (RSSI). Mobile handover in LTE-networks is usually based on these values [23] and some research on their effect on the quality of the connection during handovers has been done [3]. This research indicates that using these values to determine when to do a handover results in a better quality of connection during handovers, thus they could probably be used as a rough indication of available bandwidth for adaptive streaming as well.

3.3.1 RSRP

RSRP is a linear average over power contributions of cell-specific reference signals on a specified bandwidth which basically means that it measures the overall power received from the signal, however it gives no information of the the noise or interference levels. Its value ranges over -140 dBm to -44 dBm.[2] The higher end corresponding to a good signal quality and the lower to bad quality. Rough approximates on signal strength values corresponding to RSRP can be seen in Table 3.1.

3.3.2 RSRQ

RSRQ is calculated using formula (N * RSRP)/RSSI, where RSSI is the total power the user observers over the whole bandwidth including adjacent

RSRP (dBm)	**Signal Strength**
>-90	Excellent
-90 to -105	Good
-106 to -120	Fair
<-120	Poor

Table 3.1: Approximate signal strength values.[22]

channel interference and even thermal noise, and N is the number of physical resource blocks over which RSSI is measured. Thus, RSRQ value includes power measurements of the specific channel of the user and interference over the whole bandwidth.[2] Rough estimates of the signal quality corresponding to RSRQ values are shown in Table 3.2.

RSRQ (dB)	**Signal Quality**
>-9	Excellent
-9 to -12	Good
<-13	Fair to poor

Table 3.2: Approximate signal quality values.[22]

3.4 Bandwidth estimation

Bandwidth estimation in LTE is a problem for many applications[15]. Accurately estimating the available bandwidth is important in MCG scenario because using too little bandwidth results in bad video quality and using too much results in too high rtt for gaming purposes. However, estimating bandwidth accurately in LTE-networks is challenging, because for MCG to work in LTE, the estimates have to be fast to detect rapid changes in the network quality and the estimation technique have to be as non-intrusive as possible to keep overhead as low as possible. Bandwidth estimation has been subject of many studies and multiple algorithms exists for example Capprobe [18], Pathchirp [25] and Wbest [21]. These methods rely on sending packet trains to the network and measuring differences in delays between packet pairs. Thus, these methods are intrusive and require some time to generate the estimate. Moreover, non-fifo packet scheduling in LTE-networks might be a problem for these algorithms because packet pairs might arrive in wrong order.

3.4.1 Pathchirp

Pathchirp is an early bandwidth estimation method which send so called packet chirps to the network. Each chirp sends packets to the network with increasing intensity and measures the inter-arrival time of packets in each chirp. The maximum bandwidth is detected when the packet intensity of the chirp causes congestion in the network and the arrival times between to packets increase because of the congestion. [25]

This method introduces a lot of traffic in the network and requires almost 20 seconds convergence time [21]. Thus, the Pathchirp method is probably not the best choice for bandwidth estimation in LTE networks for MCG since fast convergence time is necessary.

3.4.2 Capprobe

Capprobe is more recent approach to bandwidth estimation which also generates its estimates by measuring the delay between packets in a packet train. Capprobe also implements packet dispersion measurements to filter out distorted by cross-traffic which makes its estimations more accurate. Moreover, it implements speed-up techniques thus it convergences faster than Pathchirp. Capprobe usually requires only a few seconds to converge.[18]

3.4.3 Wbest

Wbest algorithm improves over Capprobe by providing a detailed analytic model of packet dispersion introduced in [20]. Wbest has an average converge time of less than a second and it also introduces very little traffic to the network[21]. Thus, it seems very promising approach to estimating bandwidth in MCG scenario, however it is designed for WLAN networks and not LTE which might be a problem because of non-fifo packet scheduling in LTE.

GA implements a bandwidth estimator inspired by Wbest which removes the network overhead by leveraging the existing video packets for bandwidth estimation[11]. While this method removes the overhead, it makes the estimation slower because the video frames are sent at frame rate chosen for the video stream. This frame rate could be as low as 20 frames per second and hundreds of packets are needed to generate an accurate estimation of the bandwidth. This approach works for situations where the bandwidth stays the same for long periods of time, however in LTE-networks the bandwidth can change in a matter of seconds which probably makes this approach too slow.

Chapter 4

Adaptive video streaming

This chapter introduces the concept of adaptive streaming in a mobile gaming context and describes the differences to normal adaptive video streaming. Also, video codec h264 is introduced.

4.1 Different types of video streaming

In cloud gaming systems the cloud renders the video at least to some extent to reduce amount of processing necessary at the client, thus video streaming is an important part of cloud gaming systems. It can be classified to three categories: 3D graphics streaming[9], video streaming[8] and video streaming with post-rendering operations[27]. These three approaches differ in how much of the processing is done on the cloud and how much is left to the client.

3D-graphics streaming model captures the commands being sent to the graphics library, such as OpenGL or Direct3D. These commands are then encoded and sent to the client. Since this approach streams only the commands, it is not necessary to have a graphics processing unit (GPU) at the server and the computational demands are low. However, this means that the graphics commands have to be rendered on the client side, requiring a GPU in the mobile device. All the mobile devices does not have a GPU which makes this approach impractical in many cases. Moreover, processing the graphics commands at the mobile device requires a lot of processing and power rendering the benefits from a cloud gaming scenario very small.[9]

In the video streaming approach the 3D-graphics are rendered into a 2D-video which is compressed using video codecs, such as h264 or vp9, on the cloud server. The compressed video stream is then sent to the client for decoding and displaying. The decoding can be done using a hardware video

encoding chip, thus a GPU is not necessary at the client. Since this approach does not require a specific GPU at the client, it is more easily portable to different platforms.[8] Furthermore, it relieves the client from computationally-intensive 3D graphics rendering, thus preserving the resources of the mobile device. However, this approach requires more computation on the cloud server which might become a problem if multiple users are using the same server.

Video streaming with post rendering is a compromise between the other two approaches. The 3D-graphics rendering is still done on the cloud, however some post-rendering operations, such as augmenting motion and lightning, is left to the client. These post-rendering operation are not computationally intensive, thus they can be done without a GPU in real time. This approach holds the benefit of not needing a GPU at client while decreasing the computation necessary on the cloud. However, the post-rendering still drains the resources of the mobile client even though they are not computationally intensive. [27]

Most of the proprietary cloud gaming systems employ the video streaming approach to keep the load on the client as low as possible. GamingAnywhere, the platform used on the experimental part of this thesis, also uses this approach.[12]

4.2 Video codecs

Video codecs are either hardware or software used to decompress digital video. They are used convert raw digital video to compressed format and back to video. In cloud gaming systems with adaptive streaming, decompressing video is necessary. The codec used needs to be reconfigured on the fly to adapt to the changing network conditions thus keeping the bitrate of the video low enough for the available bandwidth. A popular video codecs H.264 and its configurability on the fly is introduced in detail in the following subsections.

4.2.1 H.264 codec

H.264 otherwise known as MPEG-4 Part 10, Advanced Video Coding is a video compression standard approved in 2003. The goal of the standard was to create a more flexible and customizible stardard than its predesessor. The flexibility was necessary because of the wide variety of network applications which needed video encoding in diffrent kinds of networks. This flexibility was implemented by a design consisting of multiple layers. A video coding

layer was designed to represent the video content, a network abstraction layer to format the video and provide heders and a data partitioning layer to partition the data. This structure is shown in Figure 4.1.[29]

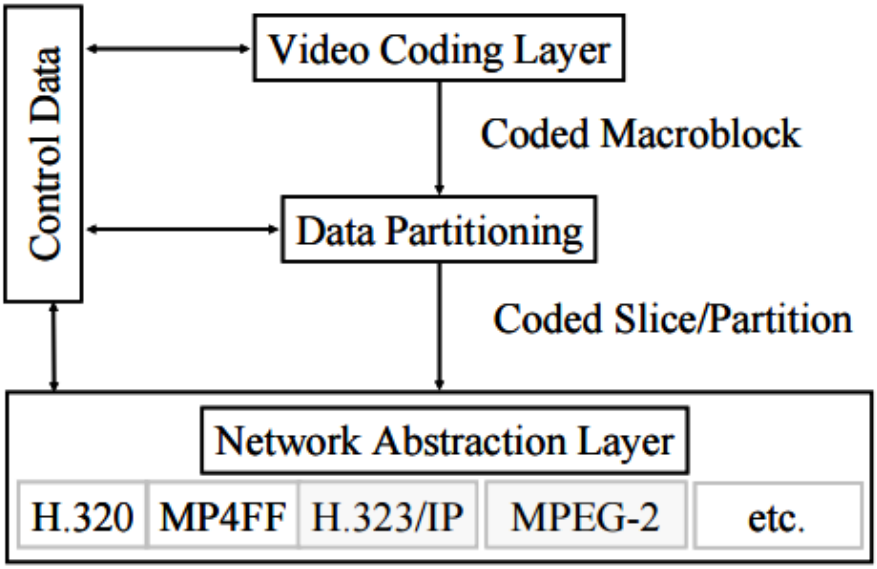


Figure 4.1: Structure of H264 video encoder.[29]

4.2.2 Reconfiguring codecs at run time

Reconfiguring codecs at runtime is necessary for adaptive streaming because it enables the server to change the bitrate of the video stream to match the current network bandwidth. Not all video codecs support reconfiguration at run time, thus selecting one that does is paramount. Video codec implementations can be either hardware-based or software-based. Software implementations are usually more flexible offering more control over the bitrate of the video stream. However, software-based implementations are much less efficent and might introduce too much delay while encoding the video stream. On the other hand hardware-based implementation are more efficent but less flexible in reconfiguration, thus selecting the right approach is not trivial.

GamingAnywhere implements three H.264 encoder modules which supports dynamic reconfiguration. First is a software-based module which uses the *libx264* library and the *x264_encoderre_config* function to reconfigure the codec. The other two are hardware-based modules which work with the video processing unit on ARM-based CPUs and the quick sync video (QSV)

technology on Intel's HD graphics.[11] The experimental part of this thesis evaluates the performance of the software-based module and uses constant rate factor (crf) parameter and quantization parameter (qp) to reconfigure the encoder on the fly.

Quantization parameter is used to determine the quntization of transform coefficents in H.264 encoder. It can have values between 0-51 and increasing the value by one means approximately 12% increase in quantization step size. This also results in a 12% reduction in the bitrate of the video.[29] The crf parameter makes use of qp parameter in reducing the bitrate of the video but it modifies its value dynamically. A higer value is chosen for parts of the video frame which have a lot of motion and a lower value for the stationary parts. Thus, using crf parameter results in subjectively better video quality because humans notice less detail in moving images than stationary images.

Chapter 5

Experiment design

This chapter describes the conducted experiments of this thesis and the modifications made to GamingAnywhere software. GA is an open source mobile cloud gaming platform which makes it an ideal testbed for the experiments in this thesis. The first experiment evaluates the applicability of GA bandwidth estimators for video stream adaptation. The second and third experiment evaluate the usefulness of the measured rtt and signal strength value for video stream adaptation. Results of these experiment are shown and discussed in Chapter 6.

5.1 Bandwidth estimator of GA

The bandwidth estimator of GA is inspired by the technique introduced in [21] with the modification that instead of packet trains the existing video packets are used for the bandwidth estimation.[11] This reduces the amount of overhead introduced to the network, however it also makes the estimation slower because the video frames are sent at fps-rate of the video. This might be a problem because in mobile networks the quality of the connection can change very rapidly. Another problem with the estimator could be that it is designed for situations where the last hop of the path is a WLAN while this thesis focuses on a situation where the last hop is a LTE-network.

The first experiment of this thesis evaluates the current bandwidth estimator of GA and its applicability to LTE-networks. For this experiment the bandwidth estimator was first tested in WLAN and LTE environments and then it was modified to provide the estimate faster by reducing the sample size used for the estimates. For this method to be useful in a situation where the quality of the connection is changing rapidly, it would have to provide an estimate at least once a second, thus the sample size is reduced to 60 frames

for this experiment. Since the frame rate will be 60 frames per second, an estimate will be provided at one second intervals.

For this experiment the GA bandwidth estimator is evaluated in WLAN and LTE networks using 3000 and 60 sample estimates as shown in Table 5.1. The first test case with WLAN as a network and 3000 sample estimates is the situation for which this algorithm is designed for. The second and third test cases might give some insight on whether the network type or the sample size is more impactful on the accuracy of the estimations. For this estimator to be usable for video stream adaptation in MCG systems the fourth test case needs to give accurate estimations of the actual bandwidth.

Network	**Samples**
WLAN	3000
LTE	3000
WLAN	60
LTE	60

Table 5.1: GA bandwidth estimator test cases

5.2 Rtt and signal strength experiments

The rtt method adapts the video stream using the rtt of the video frames as an indicator of whether the bitrate needs to be lowered or increased. The signal strength method first measures the maximum bitrates for different signal strength ranges and then uses the signal strength value to set an appropriate bitrate for the video stream. Both of these methods are evaluated on two different test cases. First is walking inside where the signal strength changes very rapidly and the second is outside where the signal strength is weak but stable. Table 5.2 summarizes the conducted experiments on these two methods and the following sections describe them in more detail.

Adaptation algorithm	**location**
Rtt based	inside
Rtt based	outside
Signal strength based	inside
Signal strength based	outside

Table 5.2: Rtt and rs based estimator test cases

5.2.1 Rtt as a bandwidth estimator

This experiment evaluates if the measured rtt value provided by RTCP-protocol can be used to adaptively reconfigure the encoder to accommodate to the changes of the available bandwidth of the connection. Although this method does not actually estimate the bandwidth, it measures the rtt value and changes the bitrate of the video by modifying the crf-parameter of the encoder. When the rtt value raises over 150ms the crf-parameter is increased, which results in lower bitrate, and decreased when rtt is under 100ms to increase the bitrate of the video. This method should find the highest possible video quality while keeping rtt value under 150ms.

5.2.2 Signal strength as a bandwidth estimator

The last experiment consists of three parts. The first part measures the produced bitrates when using different values of the crf-parameter on 30 and 60fps video streams. In the second part the maximum bitrate of video stream was measured at different signal strength values. In the third part the results were used to implement an adaptive streaming algorithm for GA where the signal strength value of the mobile device is continuously sent to the server and used to adjust the bitrate of the video. Appropriate crf values for five different signal strength ranges were determined by combining the results of the first and second part of this experiment. This algorithm is also evaluated in two scenarios similar to the earlier algorithm.

Chapter 6

Results and evaluation

This chapter shows the results of two different algorithms for adapting the quality of the video stream and some measurements of the effect of changing the crf-parameter of the video stream. Section 6.1 evaluates the bandwidth estimator of GA. Section 6.2 presents the results from an adaptive streaming algorithm which measures the rtt values between the server and the client and uses that information to adapt the quality of the video stream. Section 6.3 shows the results of similar algorithm which uses the signal strength values instead of rtt.

6.1 Bandwidth estimation of GA

The bandwidth estimator of GA was tested on WLAN and LTE networks with 2 different settings. The first was the default setting, which uses 3000 sample frames to create an estimate, and the second used only 60 sample frames so that an estimate was generated once every second. The results are summarized in Table 6.1 further explained in the following sections.

Network	**Samples**	**GA estimated bandwidth (Mbps)**	**Actual bandwidth (Mpbs)**
WLAN	3000	80	100
LTE	3000	110	50
WLAN	60	60-110	100
LTE	60	90-140	50

Table 6.1: GA bandwidth estimations on WLAN and LTE

6.1.1 Estimates with 3000 sample frames

The estimator was first tested on a WLAN with 3000 sample estimates. The WLAN had a capacity of 100 Mbps on both downlink and uplink. The estimate produced by the estimator had an average value of 80Mbps which is a reasonable estimate of the actual bandwidth of 100Mbps. This is the case for which this estimation algorithm was designed for thus a reasonable estimate was expected.

The second test was on LTE network also with 3000 sample estimates. The actual capacity of the LTE network was measured to be 50 Mbps on downlink and 38 Mbps on uplink by speedtest.net. The bandwidth estimator of GA reported an average bandwidth of 110 Mbps which indicates that it gives very poor estimates on LTE networks even when using 3000 sample frames to generate an estimate. This experiment indicates that the bandwidth estimator of GA gives very poor estimates on LTE networks.

6.1.2 Estimates with 60 sample frames

With the lower sample quantity the estimator generates an estimate once a second which is fast enough interval to actually adapt the video stream in a rapidly changing environment of LTE. The WLAN and the LTE networks were the same networks as in the 3000 frame tests. In both networks with 60 sample estimates the reported bandwidth was fluctuating rapidly over a large scale. In WLAN the range was between 60 Mbps and 110Mbps while the actual bandwidth was 100Mbps. Although the average of this range is quite good estimate of the actual bandwidth, the individual estimates are not good enough for adapting a video stream. This is because the estimates were fluctuating very rapidly even though the network stayed the same.

The last test case, with LTE network and 60 samples per estimate, would have to provide a reasonable estimates if the GA-estimator was to be used for adapting a video stream in MCG system. However, the range of estimates was 90-140Mbps while the actual bandwidth was 50 Mbps. This result shows that the estimation method used in GA is not suitable for bandwidth estimations in LTE network, thus some other method is necessary for adapting video stream in LTE networks. Sections 6.2 and 6.3 provide two alternative methods for adapting video streams for MCG systems.

6.2 Adjusting bitrate by measuring rtt values

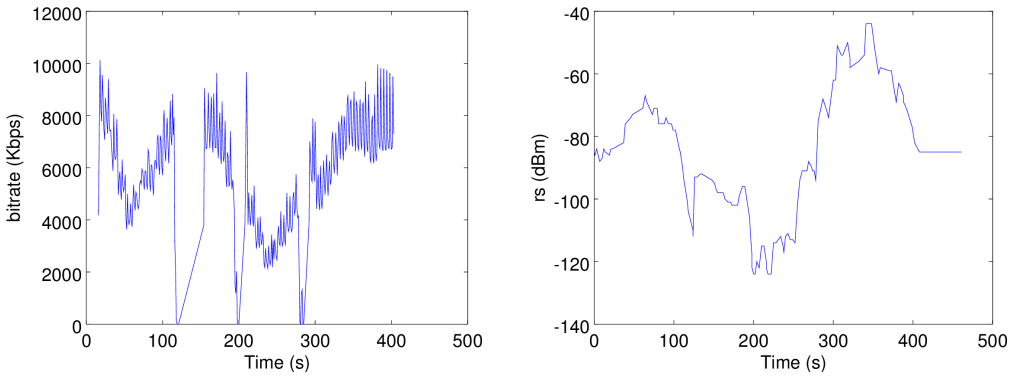
This Section shows the results from the rtt based adaptation algorithm. Section 6.2.1 shows the results from inside environment where the signal strength changes very rapidly and Section 6.2.2 shows the results from outside environment where the signal strength is weak but stable.

6.2.1 Experiments inside

Algorithm for adjusting bitrate in this test case was too slow to keep up with the changes in bandwidth of the connection. Thus, the rtt value rises really high every time the bandwidth of the connection drops. Figures 6.1,6.2 and 6.3 show the values of bitrate, rs, rsrq, rssnr and rtt during the experiment.

Figure 6.1a shows the bitrate of the video being adjusted by measuring the rtt while walking inside a building where the quality of the connection changes rapidly because of different kinds of object blocking the signal. The change in the bitrate is quite slow which often leads to overflowing buffer which shows in the figure as bitrate going down to zero. This happens because the client disconnects because of the overflowing buffer. When compared to Figure 6.1b, which shows the signal strength value, it is clear that these disconnects happen when the signal strength rapidly drops. For example, a little after the 100 second mark the signal strength drops from around -75dBm to about -110dBm. Although this is a significant drop in signal strength, it is still possible to keep some kind video stream alive at -110dBm, however in this case the client disconnects. Thus, this algorithm is adjusting the video stream too slowly.

Figure 6.2 shows the rtt value during the experiment. The first spike after the initial rtt value has stabilized starts at about 160 seconds. When compared to the rs graph in Figure 6.1b this spike is probably due to the drop in signal strength at around 140 seconds. This drop causes the rtt to stay at high value between 160s and 250s while the signal strength is really bad only between 200s and 250s. After 250s the signal strength stays at good or excellent level, however the rtt graph shows that there is still rtt spikes at around 300s and around 350s. Thus, this experiment shows that using rtt value of the video stream for adapting the quality in LTE networks is not very good solution at least in a situation where the quality of the connection changes very rapidly.



(a) Bitrate (b) Signal strength

Figure 6.1: Bitrate and signal strength while walking inside using rtt value to adapt the video stream

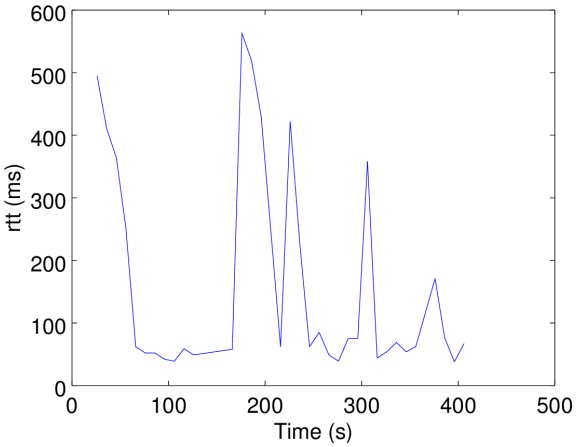
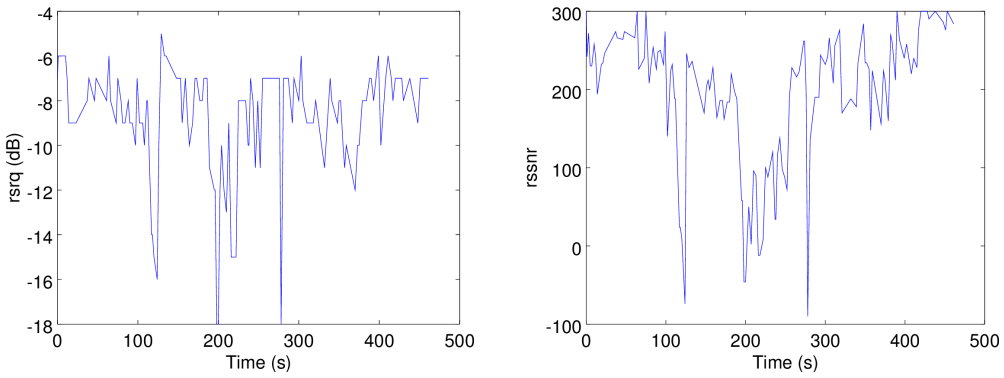


Figure 6.2: Measured rtt value

Figures 6.3a and 6.3b show the rsrq and rssnr values recorded during this experiment. Rsrq shows the interference from other channels and rssnr is the signal to noise ratio on the channel. These values are related to the rs value, thus they drop when the rs value drops, however they should provide some additional data on the quality of the connection. The graphs in these figures seem to have drops in value only when the rs value drops, thus in this case they are not very useful. This might be because there are only few users on the network which the test was conducted on so there is not much interference from other channels.

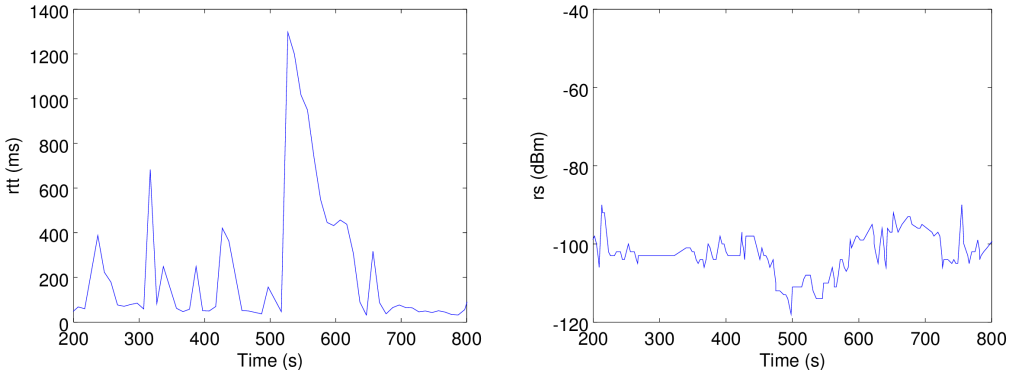


(a) Rsrq (b) rssnr

Figure 6.3: Signal strength and signal to noise ratio while walking inside

6.2.2 Experiments outside

Figures 6.4a and 6.4b show how this algorithm works in a different scenario where the signal strength is weak but it does not change very rapidly. Although Figure 6.4b shows that the signal strength value stays mostly close to -100dBm, which is weak but it should be possible to maintain a low rtt video stream when the crf value is high enough to keep the bitrate low. However, Figure 6.4a shows that the rtt value has a lot of spikes. The largest spike, which start around 510ms, happens when the signal strength value in Figure 6.4b is close to -120dBm which means that the signal strength is too weak to maintain the video stream, thus it is understandable that there is a large rtt spike at that time. However, all the other rtt spikes indicate that this algorithm does not work very well when the signal strength is weak because when the signal strength is close to -100dBm a low rtt should be maintainable.



(a) Rtt (b) Signal strength

Figure 6.4: Rtt and signal strength while walking outside using rtt to adapt the video stream

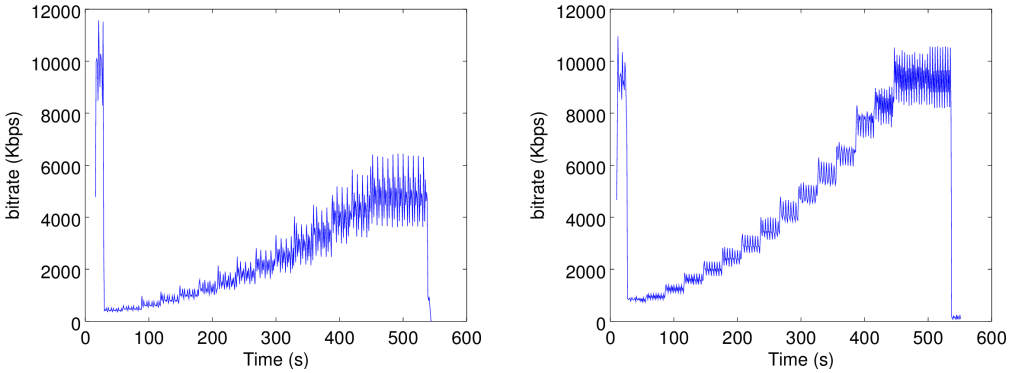
6.3 Adjusting bitrate by measuring signal strength value

This section provides an alternative algorithm for adapting the bitrate of the video by using the signal strength value of the mobile device. This algorithm requires predetermined bitrates which are matched to the signal strength values by finding the highest possible bitrate for five different signal strength levels. Section 6.3.1 shows how these bitrates were determined and Sections 6.3.2 and 6.3.3 show how this algorithm works in inside and outside environments.

6.3.1 Bitrates and crf values

The purpose of this experiment was to find appropriate bitrates for five different signal strength levels. At each signal strength threshold an algorithm was run which searched for highest possible bitrate which could be sent without rising the rtt of the video frames to higher than 150ms. The chosen values are shown in Table 6.2. The corresponding crf-values were determined by finding the appropriate bitrates from figure 6.5b, which shows the measured bitrate values of the 60 fps video while decreasing the value of crf by 3 in 30 second intervals starting from value 49. Video with 60 fps was chosen because 30 fps video does not produce high enough bitrates to test the algorithm when the signal strength is -70dBm or better. These measurements were used to develop an algorithm which adapts the video stream to the changing quality

of the mobile network by monitoring the signal strength and changing the crf value according to Table 6.2.



(a) 30fps (b) 60fps

Figure 6.5: Bitrates measured with decreasing value of crf-parameter. Crf value starting from 49 and decreasing by 3 every 30 seconds

rs (dBm)	**bitrate (Mbps)**	**crf**
-60	8	7
-70	7	10
-80	4	21
-100	3	31
-110	2	40

Table 6.2: Chosen bitrates and corresponding crf-values

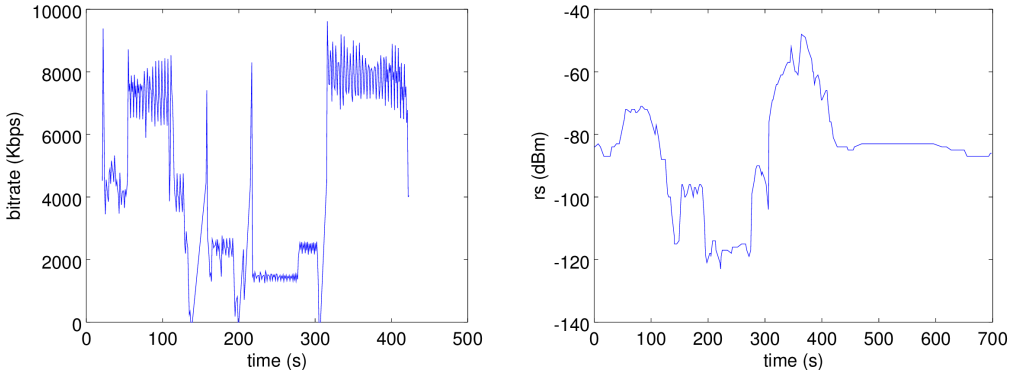
6.3.2 Experiments inside

Figure 6.6a shows the bitrate while it is being adjusted by the measured signal strength in Figure 6.6b using the values in Table 6.2. In this case the bitrate is set to predetermined values, thus the right value does not have to be searched by continuously increasing or decreasing the crf value. However, even in this scenario the client disconnects three times and the bitrate goes to zero but this is probably because the path chosen for this experiment had some locations where the signal quality was so bad that the connection was lost entirely. Despite the disconnections this algorithm set the appropriate bitrate fast and keeps it stable when the quality of the connection is good enough.

Figure 6.6b shows the recorded signal strength during this experiment. The path chosen is the same as in the earlier experiment, thus the graph looks similar. This path includes signal strengths values from the whole range from very bad to excellent and has very rapid changes, thus it is a challenging environment for adaptive streaming.

Figure 6.7 shows the rtt value during this experiment. It looks pretty similar to the corresponding rtt graph of the earlier experiment shown in Figure 6.2, however in this case the rtt value stabilizes a little faster and there is fewer spikes in it. As in the earlier experiment, the rtt is high between 200s and 270s because the signal strength is close to -120dBm, thus a low rtt is very hard to achieve for a video stream.

Overall there are rtt spikes at moments when the signal strength drops rapidly, however the rtt value stabilizes quickly below 100ms when the signal strength is -100dBm or better. Most of the rtt spikes of this test case are because of a lost connection. It is difficult to draw any conclusions from this experiment because of the unstable connection, however the important thing to notice is that when the signal strength is stable even for a moment the rtt quickly stabilizes at very low values. Thus, this algorithm finds a suitable bitrate for the current connection very quickly.



(a) Bitrate (b) Signal strength

Figure 6.6: Bitrate and signal strength while walking outside using signal strength value to adapt the video stream

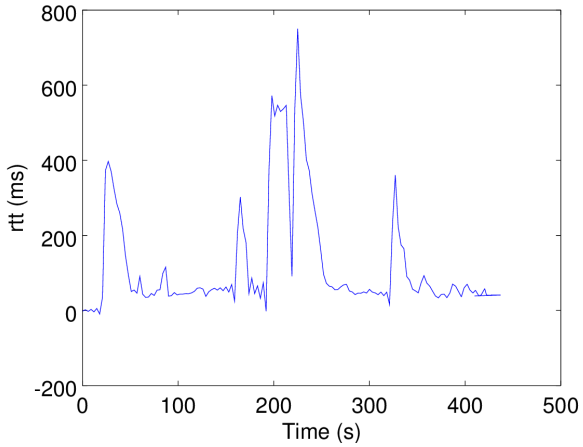
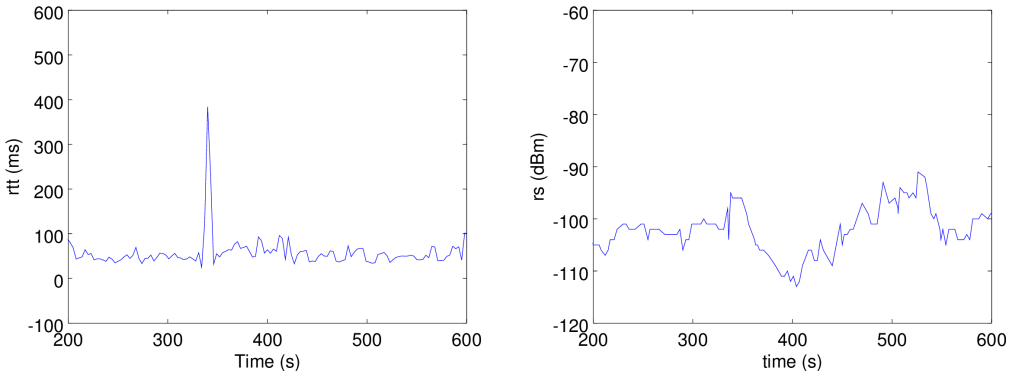


Figure 6.7: Measured rtt value

6.3.3 Experiments outside

Figures 6.8a and 6.8b show an experiment with weak but stable signal strength similar to the experiment with the earlier algorithm. As Figure 6.8b shows, the signal strength stays close to -100dBm during the experiment. Compared to the earlier experiment this algorithm performs significantly better in this scenario. The rtt value has only one spike with over 100ms rtt at around 350ms in Figure 6.8a while the rtt in the earlier experiment had multiple rtt spikes as was shown in Figure 6.4a. Thus, adapting the video stream by measuring the signal strength value works a lot better in this case than

adapting it using the rtt value as in the earlier experiment.



(a) Rtt (b) Signal strength

Figure 6.8: Rtt and signal strength while walking outside using signal strength to adapt the video stream

This test case showed that in more stable conditions, which better represent the commercial LTE network than the inside test case, the signal strength approach works better. Thus monitoring signal strength could be at least a part of a working adaptive streaming in LTE networks. However, these tests were conducted on a research network, which means that the results could be different on the commercial LTE networks. Thus, more research on this should be done to determine the usefulness of signal strength for adaptive streaming algorithms on LTE networks.

6.4 Summary

The first experiment focused on evaluating the bandwidth estimation algorithm implemented on GA. The algorithm is based on Wbest algorithm introduced in Chapter 3 and it was designed for estimating bandwidth on WLAN networks. The results showed that this algorithm produces very inaccurate results in LTE networks, thus it can not be used for adaptive streaming in LTE. The reason for the inaccurate results of the algorithm in LTE networks is probably due to the fact that Wbest algorithm relies on measuring the relative delay between packet pairs to estimate the bandwidth. This is problematic in LTE because it uses a different kind of scheduling of packets than WLAN which might lead to packets being delivered in different order than they were sent in. The algorithm is probably not designed to handle this

behavior which makes the estimations inaccurate. The results of this experiment shows that a different approach or some modification to the packet train method is necessary to estimate the bandwidth in LTE networks, because the packet train approach used in WLAN algorithms does not work in LTE networks.

The other two experiments had two test cases. The first had very rapidly changing network where the signal strength could instantly drop from excellent to very poor and the second had a stable but weak signal strength. The second test case could very well be possible in commercial LTE networks, however the signal strength of commercial LTE networks are usually more stable than the first test case had.

In first test case both of the algorithms had high rtt spikes due to the changes in signal strength during the experiment and it is difficult deduce any meaningful results from it. At worst the connection was lost completely which clearly results in an rtt spike despite the algorithm. However, in the second test case the difference in the two algorithm was clear. The rtt based algorithm had multiple rtt spikes during the whole experiment while the signal strength based algorithm only had one. Thus, the signal strength approach is clearly more suitable for adapting the video stream in MCG scenario. The advantage of the signal strength based algorithm is that it does not fluctuate because it has predefined bitrate values for different signal strength values. This however rises another problem with how well these predefined values work in different situations. For example the values chosen in this thesis work well in the Aalto Netleap LTE network but this might not be the case in commercial LTE networks. Thus, to evaluate the applicability of this adaptive streaming method, more experiments in different LTE networks would be necessary. The pros and cons of all three algorithms are summarized in Table 6.3.

Algorithm	Pros	Cons
Wbest based algorithm in GA.	Works well in WLAN.	Does not work in LTE at all. Requires a long time to get an accurate estimate.
Rtt based algorithm.	No need for any predefined parameters. Eventually finds the appropriate bitrate.	Reacts slowly to changes in the quality of the connection.
Signal strength based algorithm.	Finds the appropriate bitrate fast. Works consistently on stable conditions.	Requires a mapping of signal strength values to maximum bitrates.

Table 6.3: Pros and cons of the three algorithms

Rsrq and rssnr values were also shown as additional data in some of the experiments. Although, these values provide information on the other users on the network and interference readings on the channel, they were not useful in this experiment. The reason probably is that the network used on the experiment does not have many users, thus there was little interference during the experiments. In commercial LTE networks there often are many simultaneous users and the basestation has to share its resources. The difference in throughput in commercial LTE networks between lean and peak hour can be significant as shown in Figure 6.9. Thus including these values to the video stream adaptation algorithm could be beneficial.

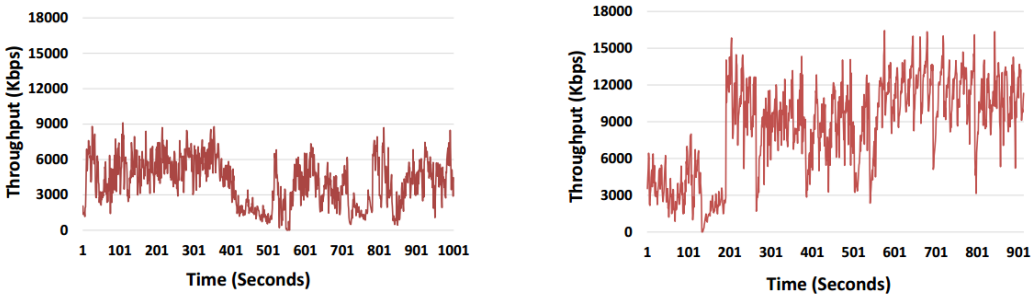


Figure 6.9: Throughput on LTE druing peak hour on the left and lean hour on the right. [1]

One approach on how to measure the current utilization of the basestation is introduced in paper[1], which proposes that the rsrq value is a good indica-

tion of the load in the cellular network, while another approach PiStream [33] monitors the resource allocation on the physical layer of LTE basestations to determine how free resources it currently has. In essence though, these approaches are similar because PiStream monitors the other channels power levels individually while rsrq represents the average power over all of them as explained in Section 3.3.2. However, either of these methods in conjunction with the signal strength monitoring method introduced in this thesis could be utilized to develop an adaptive streaming algorithm which takes into account the current utilization of the basestation.

Chapter 7

Discussion

This chapter further discusses the results from the experiments and proposes some additional experiments for future work. Additionally, possible applications for adaptive streaming in LTE networks is briefly discussed.

7.1 Adaptive streaming in LTE networks

Adaptive streaming in LTE networks is still quite new topic and it still has some unresolved problems. The most challenging problem probably is the difficulty of estimating the available bandwidth. It is especially difficult in MCG scenario, where the estimates have to be precise and fast to accommodate the rapid changes in LTE network signal strength and there cannot be long video buffers which can be filled during a good connection. Moreover, algorithms which generate very fast and accurate estimates in WLAN networks do not work very well in LTE networks because of the differences in packet scheduling. However, LTE-network architecture has much information available about the connection and utilization of the basestation which can probably be used to make accurate estimates about the currently available bandwidth. The signal strength approach presented in this thesis could be at least a part of the solution of an accurate bandwidth estimation in LTE.

7.2 Applications for adaptive streaming

There is not much information available on using the signal strength values of LTE networks for adaptive video streaming, however mobile gaming is becoming more and more popular thus methods for adaptive streaming in LTE networks are necessary. Moreover, conventional bandwidth estimation

methods, which use packet train related methods to estimate the bandwidth, work poorly on LTE networks. Thus, some other methods are necessary to implement adaptive streaming on LTE networks. Although it is challenging to implement, it would be very useful for many applications to have a accurate bandwidth estimator for LTE networks.

Adaptive streaming is clearly necessary for MCG because a low rtt video stream is required for many fast paced games. However, a low rtt video stream could also be beneficial for other applications. For example, it could be utilized in a mobile version of a graphical remote access applications, such as Chrome Remote Desktop which allows the user to graphically access a desktop computer with a mobile device. This kind of a remote control application requires a low rtt video stream to be responsive and adaptive streaming could be able to provide it even when the mobile bandwidth is not the best possible.

Chapter 8

Conclusions

Mobile cloud gaming is gaining ground as a new service for mobile devices. However, the QoS requirements for a MCG service are very demanding and they are difficult to implement because of the restrictions of mobile networks. Although LTE networks can provide short latency and enough bandwidth for MCG systems under good conditions, there still is situations where the signal is not strong enough to provide a good service. For a MCG service to work, it needs to be able to adapt to the changing network. Thus, a reliable and fast way to estimate the current bandwidth of the network is necessary.

This thesis evaluated a bandwidth estimation method used in GA. This method generates its estimates by measuring the relative delay between multiple packet pairs, which in GA are the video frames sent to the client. The differences in delay between different packet pairs can be used to measure the bandwidth in WLAN networks. However, when this method is used in LTE networks the results are not accurate. This is probably because LTE networks have a different packet scheduling system than WLAN networks, for example packets might be delivered out of order which makes measuring the delay between packet pairs difficult. Thus, bandwidth estimation methods which use this technique are not suitable for LTE networks or atleast they need some modifications to account for the packet scheduling in LTE networks.

This thesis also proposed a different approach in estimating the bandwidth of a LTE networks, which utilizes the signal strength information provided by the mobile device and the LTE base station. The experiments showed that the RS value from the mobile device can give an accurate estimation of the available bandwidth and it can be easily used to adapt the quality of the video stream in GA. This method has the benefit of not generating any overhead on the network because not extra packets has to be sent over the network. Moreover, the RS value is provided by the mobile device,

thus there is no need for any computation either. However, the experiments were done in Aalto Netleap LTE network which does not have many users, thus this method should still be evaluated under a network with more traffic and interference, to determine if the RS value still is good indicator of available bandwidth under those conditions. The LTE base stations also provide additional signal strength data, such as RSRQ and RSSNR, which could be used to make the estimation more accurate.

Bibliography

[1] *Coordinating Cellular Background Transfers using LoadSense* (September 2013).

[2] 3GPP. 3rd generation partnership project specification. `http://www.3gpp.org`. [Online; accessed 10-7-2016].

[3] Anas, M., Calabrese, F. D., Ostling, P.-E., Pedersen, K. I., and Mogensen, P. E. Performance analysis of handover measurements and layer 3 filtering for utran lte. In *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications* (2007), IEEE, pp. 1–5.

[4] Cai, W., Leung, V. C. M., and Chen, M. Next generation mobile cloud gaming. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on* (March 2013), pp. 551–560.

[5] Cai, W., Zhou, C., Li, M., Li, X., and Leung, V. C. Mcg testbed: An experimental test-bed for mobile cloud gaming. In *Proceedings of the 2Nd Workshop on Mobile Gaming* (New York, NY, USA, 2015), MobiGames '15, ACM, pp. 25–30.

[6] Chen, K.-T., Chang, Y.-C., Tseng, P.-H., Huang, C.-Y., and Lei, C.-L. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM International Conference on Multimedia* (New York, NY, USA, 2011), MM '11, ACM, pp. 1269–1272.

[7] cluster Global Corporation, G. Square Enix Japan to Launch Streaming Service Using G-cluster Technology-1 Targeting Smartphones and Tablets. `http://http://www.gcluster.com/eng/pdf/20140909_001_E.pdf`, 2014. [Online; accessed 17-4-2016].

[8] De Winter, D., Simoens, P., Deboosere, L., De Turck, F., Moreau, J., Dhoedt, B., and Demeester, P. A hybrid thin-client

protocol for multimedia streaming and interactive gaming applications. In *Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video* (2006), ACM, p. 15.

[9] Eisert, P., and Fechteler, P. Low delay streaming of computer graphics. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on* (2008), IEEE, pp. 2704–2707.

[10] H. Schulzrinne, A. Rao, R. L. Real time streaming protocol (rtsp). rfc 2326 (proposed standard). `http://www.ietf.org/rfc/rfc2326.txt`, 1998. [Online; accessed 25-4-2016].

[11] Hong, H.-J., Hsu, C.-F., Tsai, T.-H., Huang, C.-Y., Chen, K.-T., and Hsu, C.-H. Enabling adaptive cloud gaming in an open-source cloud gaming platform. *IEEE Transactions on Circuits and Systems for Video Technology 25*, 12 (Dec 2015), 2078–2091.

[12] Huang, C.-Y., Chen, K.-T., Chen, D.-Y., Hsu, H.-J., and Hsu, C.-H. Gaminganywhere: The first open source cloud gaming system. *ACM Trans. Multimedia Comput. Commun. Appl. 10*, 1s (Jan. 2014), 10:1–10:25.

[13] Huang, C.-Y., Hsu, C.-H., and Chen, K.-T. Gaminganywhere: An open-source cloud gaming platform. *SIGMultimedia Rec. 7*, 1 (June 2015), 3–5.

[14] Huang, J., Qian, F., Guo, Y., Zhou, Y., Xu, Q., Mao, Z. M., Sen, S., and Spatscheck, O. An in-depth study of lte: effect of network protocol and application behavior on performance. In *ACM SIGCOMM Computer Communication Review* (2013), vol. 43, ACM, pp. 363–374.

[15] Huang, J., Qian, F., Guo, Y., Zhou, Y., Xu, Q., Mao, Z. M., Sen, S., and Spatscheck, O. An in-depth study of lte: effect of network protocol and application behavior on performance. In *ACM SIGCOMM Computer Communication Review* (2013), vol. 43, ACM, pp. 363–374.

[16] Jarschel, M., Schlosser, D., Scheuring, S., and Hoßfeld, T. An evaluation of qoe in cloud gaming based on subjective tests. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on* (June 2011), pp. 330–335.

[17] Jarschel, M., Schlosser, D., Scheuring, S., and Hossfeld, T. An evaluation of qoe in cloud gaming based on subjective tests. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on* (2011), IEEE, pp. 330–335.

[18] Kapoor, R., Chen, L.-J., Lao, L., Gerla, M., and Sanadidi, M. Capprobe: a simple and accurate capacity estimation technique. *ACM SIGCOMM Computer Communication Review 34*, 4 (2004), 67–78.

[19] Lampe, U., Hans, R., and Steinmetz, R. Will mobile cloud gaming work? findings on latency, energy, and cost. In *Proc. of IEEE International Conference on Mobile Services (MS13)* (2013), pp. 960–961.

[20] Li, M., Claypool, M., and Kinicki, R. Packet dispersion in ieee 802.11 wireless networks. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks* (2006), IEEE, pp. 721–729.

[21] Li, M., Claypool, M., and Kinicki, R. Wbest: A bandwidth estimation tool for ieee 802.11 wireless networks. In *2008 33rd IEEE Conference on Local Computer Networks (LCN)* (2008), IEEE, pp. 374–381.

[22] Mammen, S. Making sense of signal strength/signal quality readings for cellular modems. `http://blog.industrialnetworking.com/2014/04/making-sense-of-signal-strengthsignal.html`. [Online; accessed 10-7-2016].

[23] Östling, P.-E. Performance of rss-, sir-based handoff and soft handoff in microcellular environments. In *Wireless personal communications*. Springer, 1996, pp. 147–158.

[24] Piro, G., Grieco, L. A., Boggia, G., Capozzi, F., and Camarda, P. Simulating lte cellular systems: an open-source framework. *Vehicular Technology, IEEE Transactions on 60*, 2 (2011), 498–513.

[25] Ribeiro, V. J., Riedi, R. H., Baraniuk, R. G., Navratil, J., and Cottrell, L. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and active measurement workshop* (2003).

[26] Schulman, A., Navda, V., Ramjee, R., Spring, N., Deshpande, P., Grunewald, C., Jain, K., and Padmanabhan, V. N. Bartendr: a practical approach to energy-aware cellular data scheduling. In

Proceedings of the sixteenth annual international conference on Mobile computing and networking (2010), ACM, pp. 85–96.

[27] Shi, S., Hsu, C.-H., Nahrstedt, K., and Campbell, R. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proceedings of the 19th ACM international conference on Multimedia* (2011), ACM, pp. 103–112.

[28] Verbelen, T., Simoens, P., De Turck, F., and Dhoedt, B. Cloudlets: bringing the cloud to the mobile user. In *Proceedings of the third ACM workshop on Mobile cloud computing and services* (2012), ACM, pp. 29–36.

[29] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology 13*, 7 (2003), 560–576.

[30] Winkler, S., and Faller, C. Audiovisual quality evaluation of low-bitrate video. In *Electronic Imaging 2005* (2005), International Society for Optics and Photonics, pp. 139–148.

[31] Wu, J., Yuen, C., Cheung, N.-M., Chen, J., and Chen, C. W. Enabling adaptive high-frame-rate video streaming in mobile cloud gaming applications. *IEEE Transactions on Circuits and Systems for Video Technology 25*, 12 (Dec 2015), 1988–2001.

[32] Wylie-Green, M. P., and Svensson, T. Throughput, capacity, handover and latency performance in a 3gpp lte fdd field trial. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE* (Dec 2010), pp. 1–6.

[33] Xie, X., Zhang, X., Kumar, S., and Li, L. E. pistream: Physical layer informed adaptive video streaming over lte. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (2015), ACM, pp. 413–425.