A Measurement Study on Achieving Imperceptible Latency in Mobile Cloud Gaming

Teemu Kämäräinen
Aalto University
Finland
teemu.kamarainen@aalto.fi

Matti Siekkinen
Aalto University
Finland
matti.siekkinen@aalto.fi

Antti Ylä-Jääski
Aalto University
Finland
antti.yla-jaaski@aalto.fi

Wenxiao Zhang
The Hong Kong University of Science and Technology
Hong Kong
wzhangal@stu.ust.hk

Pan Hui
The Hong Kong University of Science and Technology
Hong Kong
panhui@cse.ust.hk

ABSTRACT

Cloud gaming is a relatively new paradigm in which the game is rendered in the cloud and is streamed to an end-user device through a thin client. Latency is a key challenge for cloud gaming. In order to optimize the end-to-end latency, it is first necessary to understand how the end-to-end latency builds up from the mobile device to the cloud gaming server. In this paper we dissect the delays occurring in the mobile device and measure access delays in various networks and network conditions. We also perform a Europe-wide latency measurement study to find the optimal server locations and see how the number of server locations affects the network delay. The results are compared to limits found for perceivable delays in recent human-computer interaction studies. We show that the limits can be achieved only with the latest mobile devices with specific control methods. In addition, we study the expected latency reduction by near future technological development and show that its potential impact is bigger on the end-to-end latency than that of replication of the service and server placement optimization.

CCS Concepts

•Information systems → Multimedia streaming;
•Networks → Network measurement; Cloud computing;

Keywords

Cloud gaming; latency; network measurements

1. INTRODUCTION

Cloud gaming is a recently emerged application that combines the concepts of cloud computing and online gaming. It allows the end-user device to offload computation, storage, and the tasks of graphic rendering to the cloud. This reduces hardware requirements for the end-user device, as only a thin client that takes care of video decoding and user interaction is running on the device. The resulting client-side resource requirements are constant regardless of the game being played.

The main challenge for cloud gaming is latency. The user experience when playing certain games is very sensitive to latency. These games become unplayable if the delay between the action, such as tapping a control key to steer a car left, and response to that action, i.e., observing the car to turn left on display, becomes too high. The latency challenge is amplified with wireless mobile networks which typically add a longer part to the end-to-end latency compared to fixed wired access network technologies. To cope with the latency challenge, the cloud gaming system must be distributed so that the gaming servers can be brought closer to the user when necessary.

This paper is our first attempt to answer the following question: "Is it possible to implement and deploy a mobile cloud gaming service with which the user cannot perceive the latency imposed by remote computation, and if so, how?" The fact that humans cannot perceive very short latencies motivates our study. In particular, earlier work on human-computer interaction has characterized perceivable latencies in direct and indirect interaction by human with a touch screen [11]. In this paper, we study the achievability of latencies below these perceivable latency bounds in mobile cloud gaming using currently available technologies. We also do some projections based on technological advances expected in the near future.

We first show a set of measurements to characterize the range of expected latencies in mobile cloud gaming systems. We then study the effect of server deployment in the case of Europe-wide deployment of a cloud gaming service where we leverage measurements with a wide range of possible client and server locations from Planetlab[28] and Speedtest[34]. Specifically, we study the expected latencies observed by clients in different geographic locations when distributing servers in different number of optimally or near optimally chosen locations.

The problem of optimal server placement given constraints and an optimization target, such as cost or latency mini-

MMSys '17 June 20–23, 2017, Taipei, Taiwan
© 2017 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

mization, has been studied earlier with peer-to-peer systems and content distribution networks[36, 29]. Gaming imposes specific latency constraints and workload and, consequently, a few papers on cloud gaming server placement also exist[14, 35]. To the best of our knowledge, our work is the first of its kind in two ways: 1) We focus specifically on *mobile* cloud gaming and 2) we study the achievability of *imperceptible* latency in such a system. Hence, our work differs from previous studies that have used latency limits whose effect on user experience have been characterized through Mean Opinion Score (MOS) scale ratings in subjective experiments and that do not correspond to true end-to-end latencies in vast majority of cases. We argue that imperceptible end-to-end latency is the holy grail that such systems should strive for and, therefore, we feel that this study is timely and important. In addition, while previous work focuses on optimizing existing cloud gaming systems and is constrained by their server and client locations, our study is not tied to any deployed gaming system.

In summary, the contributions of this paper are the following:

- We dissect the end-to-end latency in mobile cloud gaming and study the impact of different factors on the total delay.
- We show that it is possible in mobile cloud gaming systems to achieve latencies so short that users cannot perceive them with current technologies. However, it requires distribution of the game servers, latest mobile device models, and specific control methods. Specifically, the use of touch screen adds too much latency with currently available mobile devices and, thus, prohibits reaching short enough latencies.
- We quantify the relative impact of different mobile device, server, and network induced delays and show the impact of server placement optimization for the end-to-end latency in a Europe-wide case study. We also make projections on the impact of near future technological developments on the latency.

2. BACKGROUND

2.1 Mobile Cloud Gaming

Moving games into the cloud enables users to play compute-intensive games on mobile devices like smartphones and tablets, as heavy processing is offloaded to the cloud. A dozen or so companies are offering varying cloud gaming solutions at the moment, while many trials have ended in bankruptcy or discontinuation of the service. Nvidia's Geforce Now and Sony's Playstation Now are some of the biggest cloud gaming providers at the moment with Geforce Now having support for their own mobile device although their current focus has been shifted for console gaming.

So far the most common usage of cloud computing in gaming is to deliver the game software and to manage the user data. Meanwhile, a lot of game providers have already been running multi-player game servers on the cloud. These game servers can handle the state changes of the connected game clients, while leaving the graphic rendering on the game clients. Alternatively, the game servers can take care of graphic rendering as well to keep the game clients as thin as possible, which is the focus of our work too.

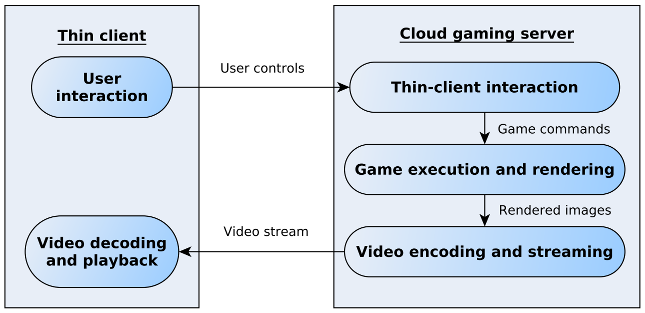

Figure 1: Framework of a cloud gaming platform (adapted from [32]).

The main advantages of moving games to the cloud are the far less strict requirements for the client hardware and the possibility to remove almost all architectural limitations. Potentially millions of new devices such as tablets and phones could gain access to games, which would run normally only on high-end desktop PCs. Users would not need to upgrade frequently their devices to support the latest games anymore and possibly enable the users to play more games thanks to the lowered hardware/software costs. As discussed in [33] and [15], cloud gaming also has potential to reduce power consumption and production costs and increase net revenues for the developers by not having to develop for several platforms at the same time.

Figure 1 presents the framework of a typical cloud gaming platform. The game is entirely executed and rendered by the cloud gaming server running in the cloud. There are two data flows between the cloud gaming server and the client. The video data flow carries rendered, captured, and encoded images from the server to the game clients which are then decoded and played back to the user. The control flow is used by the clients to capture user's control input and convey it to the cloud gaming server which replays the input on the host machine.

Similar to the work presented in [15], we term the time difference between a user's command input and the corresponding in-game action appearing on the screen as the response delay (RD). It is a key factor that affects the quality of user experience while playing cloud-based games. Following the terminology presented in [8], it can be further divided into three components: processing delay (PD), network delay (ND) and playout delay (OD).

The processing delay is the time interval between the server receiving a command from the user and submitting the corresponding video frame to the user. Network delay is the round-trip time (RTT) between the thin client and the cloud gaming server and is equal to the time it takes for a command to traverse through the network and a frame to come back to the client. Playout delay is the time it takes for the client to display a frame to the user. It includes frame reception, frame decoding and displaying the decoded frame on the display of the mobile device. In addition, the pipeline includes also control delay (CD), which means the time difference between the user initiating a command (e.g. touching the screen, pressing a button on a gamepad) and the device registering the input. Playout delay and control delay form the device delay, which together with network and processing delays create the end-to-end pipeline.

2.2 Related Work

While deep understanding of the impact of latency on user experience is still an open problem, many discoveries have been made about human perception of latency with mobile technology. Deber et al. recently characterized the *Just Noticeable Difference (JND) threshold* to a reference latency of 0.98 ms, i.e. virtually zero latency, through psychophysical experiments based on the adaptive staircase procedure[11]. They also investigated the impact of additional latency on task performance in direct and indirect user interaction with a touch device. They found that the mean JND threshold for a simple tapping task is 69 ms and 96 ms for direct and indirect touch, respectively. The threshold is substantially shorter when performing a dragging task. Lee et al. studied error rates in pointing tasks where a target is about to appear within a limited time window for selection[21].

The impact of network latency on the quality of gaming experience has been discussed in several studies. For example, Pantel et al. [27] proposed that the delay should not be longer than 100 ms based on the measurement of two racing games. On the other hand, Lee et al. [24] show that the impact of latency on quality of experience depends on the the game type because there are notable differences between the amount of screen changes in response to a player's commands. However these studies examined only the impact of network delay on traditional multiplayer servers where the delay can often be compensated. Jarschel et al. have researched the perceived QoE of users in different network conditions specifically in cloud gaming [18]. They concluded that in fast-paced games the delay component becomes the dominant metric affecting the QoE. Unfortunately, the base delay of the system was not measured in the study. This is why the latency limits found cannot be directly used to set boundaries to acceptable end-to-end latencies.

Deploying cloud services in a geographically distributed manner has also been discussed by for example Zhang et al.[37], who focused on the optimization algorithms that determine where to deploy the services with minimum cost and potentially optimal quality of service. In addition Hong et al. have studied the efficient consolidation of multiple cloud gaming servers on a physical machine[13]. In this paper we show how the network latency is affected as more server locations are added to the system. Satyanarayanan et al. have presented Cloudlets that offer computing power for mobile clients within one-hop latency[31]. The solution proposed by Choy et al. is to use the existing CDN network to offload computation from the mobile device[9]. Processing delay on the server side as well as on the client side occur both in the software and hardware. Jain et al. focus on balancing the network and the computational delay with accuracy in mobile AR[17]. Lee et al. developed a system to speculatively execute different possible scenarios of a cloud game in order to mask latency[22]. This however requires support from the game engine and results to more overhead in the transmission. Boos et al. applied the same approach to VR[4]. Their system aggressively precomputes and caches all possible images that a VR user might encounter in order to achieve low latency and energy consumption.

Concerning latency measurements, we performed a dissection of the end-to-end latency particularly on the mobile device side in our earlier work [2]. Related studies have mainly used timing hooks injected into the code or a high-speed camera[7, 16, 6]. Our method includes a modified and extended version of the WALT Latency Timer [20] together with code injections allowing a full dissection of the delay pipeline. Similar measurement setups have been previously utilized successfully in measuring mobile phone display responsiveness[3, 10]. Our approach can also measure gyro, gamepad and Bluetooth delay on top of the traditional touch latency measurements. A predictive approach has also been proposed by Cattan et al. but it requires separate calibration[5]. We summarize the latency measurement results in Section 3 and use the found base latencies in evaluating the present and future potential of providing a completely unnoticable delay in the case of mobile cloud gaming. However, the latency study for the wired segment portion of the end-to-end latency can also be utilized in designing a non-mobile cloud gaming service.

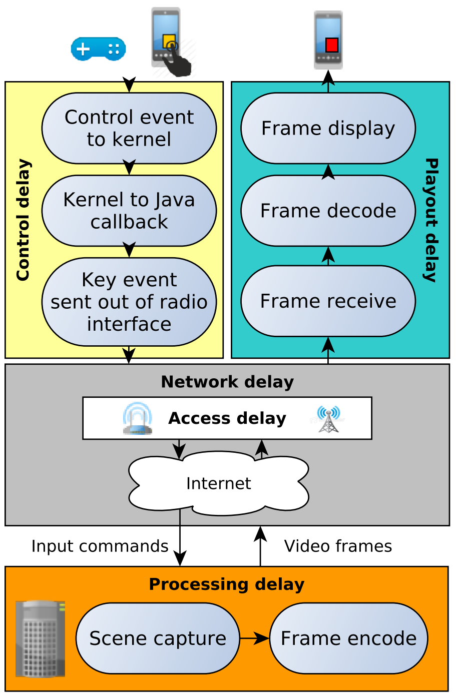

Figure 2: The complete end-to-end response delay pipeline in cloud gaming with the used terminology.

3. LATENCY MEASUREMENTS

In order to understand whether imperceptible latency is achievable, we first need to understand the end-to-end latency resulting from the current technology used in mobile cloud gaming. The complete end-to-end pipeline of the total delay together with the used terminology is presented in Figure 2. In this section we summarize our mobile device latency measurements from our previous work[2], measure the effect of different access networks and network conditions on the latency, and confirm server delay results from previous studies with our own experiments. In Section 4, we continue the analysis by performing a server placement optimization study based on the wired segment part of the network delay.

3.1 Mobile device

For analyzing the mobile device induced delays we run a number of test cases in which we vary parameters listed

Table 1: Parameters varied in the different test cases

Control	Touch / Gamepad (USB/BLE) / Gyro
Mobile phone	Samsung S4 / Samsung S7

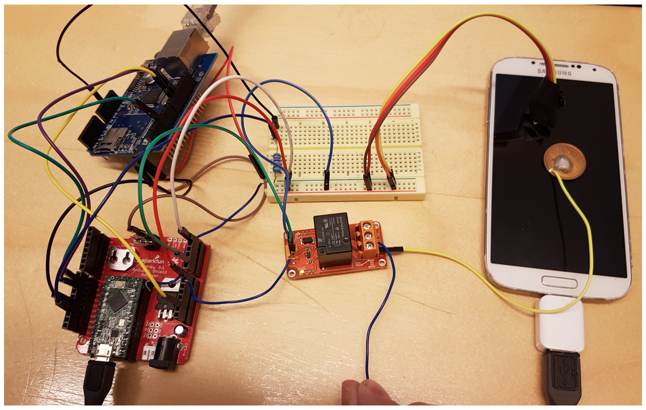

Figure 3: Touch, gamepad, Ethernet and screen delay measurement setup.

in Table 1 to find the lower bounds for the delays present in different control scenarios. The mobile devices used for the measurements are Samsung S4 and Samsung S7. The devices are equipped with the newest Android versions available for each device at the time of writing, which are Android 5.0 for the S4 and Android 6.0 for the S7. We control the device either by using the touch screen or an external gamepad connected to the mobile device both with USB and Bluetooth connection. We also measure the delay when the game is controlled using the embedded gyroscope of the mobile device. The gaming platform consists of a remote server and an Android mobile client device, both of which run the GamingAnywhere software[15].

3.1.1 Control delay

We define control delay in the mobile device to be the delay between user initiating a control input and the control command to be sent through the radio interface of the mobile device. The most common interface for user commands is a virtual gamepad using the touch screen of the mobile device. However, an external or embedded USB-connected gamepad can also be used in mobile cloud gaming. The gamepad can also be attached via a wireless Bluetooth connection. Additionally, the gyro sensor inside the mobile device can be used for controlling certain games. Each control method introduces different amounts of delay to the pipeline.

In our previous work we have measured the control delay of the input options mentioned previously using an Arduino-compatible measurement device depicted in Figures 3 and 4. The device is based on the WALT[20] project. The device is built around the Teensy 3.2 USB development board. The device can trigger a touch input on the mobile phone using a coin attached to a relay. The device can also act as a gamepad inputting control commands through the USB interface. The two attached photodiodes can detect the change in illumination of a marked frame on the display of the mobile device. This information together with time synchronization with the mobile device is used to calculate the base latencies of the mobile devices. The time error between the devices is minimized using the USB connection from the Teensy board to the mobile phone before each experiment. The setup is capable of synchronizing the clocks within 100μs accuracy

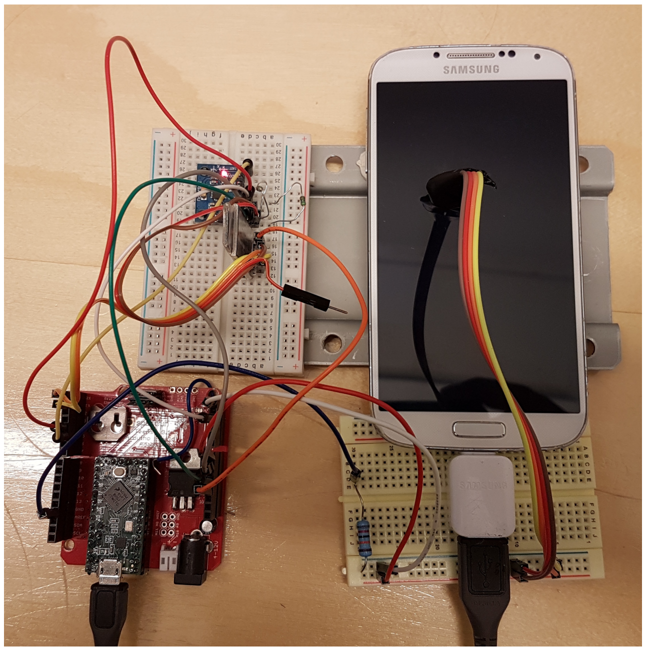

Figure 4: Gyro and Bluetooth delay measurement setup.

Table 2: Control delay measurement results.

	Samsung S4		Samsung S7	
	Avg.	**SD**	**Avg.**	**SD**
Touch to kernel (ms)	40.5	2.3	24.1	3.0
Gamepad to kernel (ms)	0.6	0.6	0.2	0.4
Kernel to callback (ms)	5.5	1.6	3.4	0.6
Callback to radio (ms)	9.1	2.7	1.6	0.8

In addition we measured the delay of the embedded gyroscope in the mobile device by comparing it to raw readings from a gyro attached to the measurement device. Both the reference gyro and the mobile phone were attached to a metal plate. When moved, both create a similar acceleration trace. Comparing the start and end times of movements, we could measure how much delayed are the gyro readings available for mobile application. The measurement device was also equipped with a BLE (Bluetooth Low Energy) chip, allowing us to measure the delay in receiving commands through a Bluetooth connection.

The average delays and standard deviations are summarized in Table 2. The results show the delays from the user input to the kernel registering the event and the delay from the kernel event to the application code (callback). We also measured how long does it take to prepare a single control event packet and send it through the radio interface. Touch screen has the slowest response on both tested devices. The delay however varies signifigantly between the devices. The newer Samsung S7 averages at 29.1 ms total control delay while the older Samsung S4 uses 55.1 ms on average for processing the touch input. A USB-connected gamepad is the

Table 3: Mobile device gyro sensor and Bluetooth delay.

	Samsung S4		Samsung S7	
	Avg.	SD	Avg.	SD
Gyro delay (ms)	78.8	32.8	12.2	4.1
BLE delay (ms)	17.5	5.2	22.0	4.9

Table 4: Frame receive, decode and display measurement results.

	Samsung S4		Samsung S7	
	Avg.	SD	Avg.	SD
Frame receive (ms)	10.5	5.8	9.6	4.5
Frame decode (ms)	20.4	11.6	8.3	1.1
Frame display (ms)	25.1	5.4	27.3	4.7

fastest input method on all devices ranging from 5.2 to 15.2 ms.

For the Bluetooth and Gyro delays, we were only able to measure the complete delay from input to the application callback. The results are shown in Table 3. Bluetooth (BLE) response varies between 17.5 ms and 28.2 ms on the tested devices. Gyro response is fast, around 10 ms for the more newer devices. The gyro sensor on the Samsung S4 however seems to be significantly slower with a delay of 79 ms.

In the cloud gaming use case the controls are sent directly to the cloud gaming server. We measured this delay to be 9.1 ms on the Samsung S4 and 1.6 ms on the Samsung S7. Overall the results show the decrease in control delay using the more recent mobile device and Android operating system version.

3.1.2 *Playout delay*

In order to measure the total playout delay (OD) perceived by the player, we slightly modified the GamingAnywhere Android client by injecting timestamps to different parts of the code. In this way, we could log and analyze also the breakdown of the total OD into delay caused by frame reception and frame decoding. We measured the timestamping code to add under 1 ms of additional latency to the system. The measurement is repeated for several frames processed giving us the possibility to calculate the averages over a time period.

Frame receive and decode: The cloud gaming client receives a single frame in multiple network packets and the entire frame is then decoded using the hardware accelerated decoders. Using the GamingAnywhere client, we measured both delays with 1080p resolution. The results are presented in Table 4. The results show that the delay of frame reception is not dependant on the device. However, frame decoding is substantially faster with the newer Samsung S7 which decodes a single frame in roughly 8 ms compared to the 20 ms of the Samsung S4.

Frame display: After decoding the frame is sent to the display buffer. We define frame display delay as the time between the frame returning from the video decoder to the time the frame is displayed on the screen of the mobile device. Our previous measurements have shown that this final delay is a substantial addition to the overall base latency. Table 4 includes the frame display times measured for each of the mobile phones. The frame display time does not vary substantially between the tested devices. This is because all phones have similar displays with 60 Hz refresh rates which translates to the screen updating roughly every 17 ms. The results however show an average of 1.5 display refresh periods from decoder output to the frame being visible on the screen. This is because Android uses double buffering to avoid screen tearing with the expense of display delay.

3.2 Network delay

The cloud gaming client is connected to the Internet using either a WiFi or a dedicated campus LTE network which was very lightly loaded. The deployment includes two cases: When using WiFi access, we deploy it in the same local network as the mobile device. When using the LTE access, the gaming platform is deployed behind a fibre connection from the Internet Service Provider (ISP) which in practice provides similar latency as if it is deployed within the ISP's network. We measured a difference of 1-2 ms when measuring the latency to our gaming platform server compared to the first pingable IP address behind the packet core of the LTE network from our mobile device.

3.2.1 *Access delay*

Access delay is the latency between the mobile device and the first pingable IP address. In the Wi-Fi use case this is the access point and in the case of mobile networks it is a machine just behind the packet core. Regardless of the distance between the cloud gaming client and the server, access delay must be accounted for to the base latency of the system. We connect the tested mobile devices both to a dedicated campus LTE network and to a local network through a Wi-Fi router. The LTE network is a full-fledged network setup with a maximun download speed of 70 Mbps (category 3) with practically no load. The average delay for the LTE scenario was 12 ms. The Wi-Fi network was faster in optimal conditions with an average delay of only 1.6 ms.

3.2.2 *Impact of Access Network Conditions on Latency and Throughput*

In the optimal test cases the LTE network has virtually no load and the nearest base station is located close within the same building as the test device. In reality the signal strength of the connection might vary drastically between geographical locations which affects the response delay. We next look at the effect of signal strength and cross-traffic to the delay of the LTE scenario.

We measure the effect of signal strength on the network delay by logging the LTE Reference Signal Received Power (RSRP) and the latency to the closest pingable IP address while walking closer and further from the base station. RSRP is defined as the linear average over the power contributions of the resource elements that carry cell-specific reference signals within the measurement frequency bandwidth [1]. Figure 5 highlights the effect of signal strength for the delay.

We observe three distinct patterns in the delay. The delay is really stable from -50 to just under -70 dBm. After that the fluctuation of the delay becomes more noticeable up to a signal strength of -100 dBm. Starting from just under -100 dBm, the delay starts to fluctuate more heavily and the average delay doubles quickly. At -100 dBm the delay has already doubled and around -110 dBm the average delay is four times the delay in the optimal conditions and continues to grow exponentially. We used packet length of 1700 bytes in the experiments which we measured to be the aver-

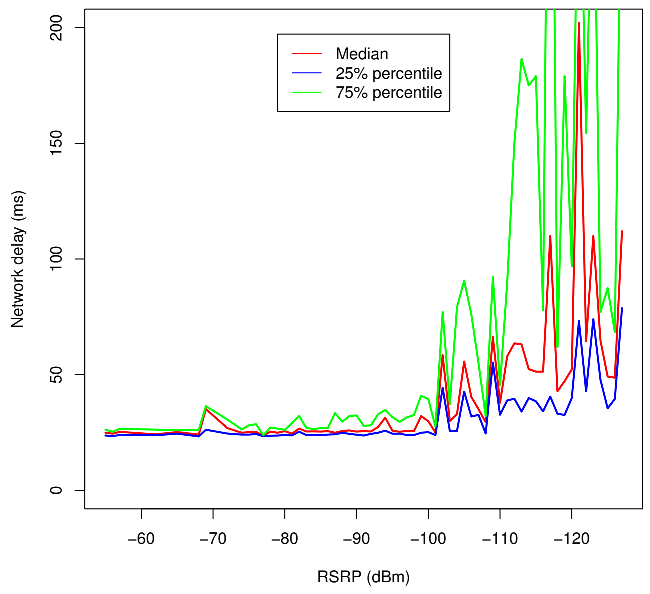

Figure 5: The effect of signal strength on delay measured to the first pingable IP address in the LTE network.

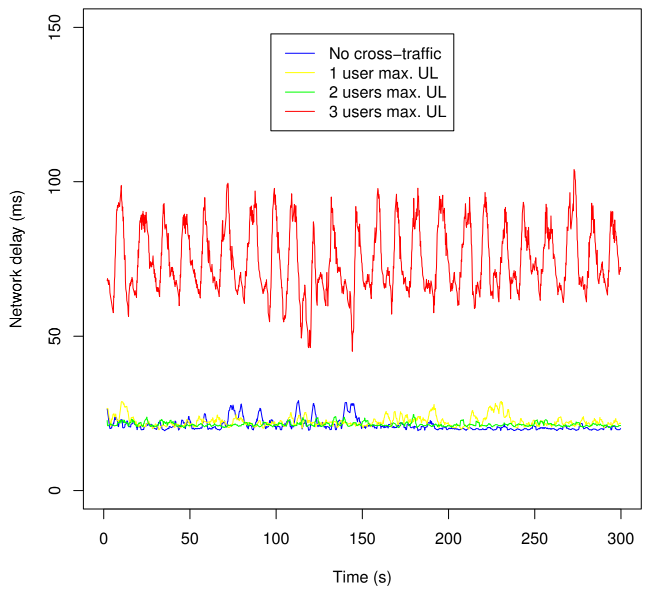

Figure 6: The effect of uplink traffic on the moving average of delay measured to the first pingable instance in an LTE network.

age packet length of the RTP traffic used between the cloud gaming client and server. Total of 3000 latency/RSRP pairs were measured. The packet loss in the link layer is the main contributor for the delay in the worst signal strength conditions. Such low signal strength values are however rarely observed in commercial network with decent network coverage. In our further analysis of the delay results in Section 4 we presume a network with good coverage and thus don't take into account the increased network delays in extreme conditions.

The traffic load in the access network also affects directly the delay perceived by the user as the user's mobile phone shares the available resources with others. We were able to create a notable difference in the delay with three other mobile devices connected to the base station. We uploaded random data with Iperf to a server with one to three LTE category 4 mobile phones while measuring the latency to the nearest pingable instance with an LTE category 3 mobile phone. Using only one or two devices to create traffic was not enough but with three devices sending traffic uplink at their maximum rates, the test device's latency started to fluctuate. Figure 6 shows the fluctuation with one to three mobile phones creating uplink traffic. We did not observe similar results in downlink traffic. However increasing the amount of users generating traffic should worsen the delays observed also in the downlink case. The uplink scenarios are however rare with current mobile applications. The majority of the traffic is downstream with the exception of some video broadcasting applications. Even those don't however consume the entire uplink bandwidth.

3.2.3 *Wide Area Network (WAN) delay*

WAN delay depends on the location of the cloud gaming server and the distance between the user and the server. We conducted a ping latency study between a large number of hosts within Europe in order to characterize the latency contribution of the wired segment of the network path. We used the Planetlab Europe testbed to measure the latency from 79 Planetlab servers to 1622 speedtest.net server locations. Both the Planetlab and the Speedtest.net servers are geographically distributed across Europe. The servers included in the study are displayed in Figure 7.

Next, we divided the European continent into smaller geographical areas based on the Nomenclature of Territorial Units for Statistics (NUTS) classification of areas provided by Eurostat[12]. NUTS is a hierarchical system for dividing Europe into geographic territories for regional statistics analysis. We chose the smallest (level 3) areas for consideration and calculated the mean average latencies to each area from the 79 Planetlab servers.

We analyze the delay benefits of distributing the cloud gaming architecture based on the gathered latency data in Section 4.

3.3 Processing delay

The cloud gaming server captures the game scene, encodes it into a video stream and sends it to the cloud gaming client. Simultaneously the cloud gaming server receives input commands from the client and feeds them to the game through the operating system. The processing delay depends on the specific hardware and software used for the cloud gaming server. Huang et al. [15] benchmarked the open-source cloud gaming software GamingAnywhere against commercial alternatives available at that time on a desktop PC. GamingAnywhere had the lowest delay ranging between 27 and 34 ms depending on the game.

This delay can however be shortened by using purpose-built features of modern graphics cards. Nvidia's solution is to capture the framebuffer on the fly and hand it to the built-in NVENC chip which is capable of encoding the graphics output in realtime. These features are available for cloud gaming providers through the Capture SDK[25] and is supported on selected graphics cards. We launched an Amazon G2 instance with an Nvidia GRID K520 graphics card and

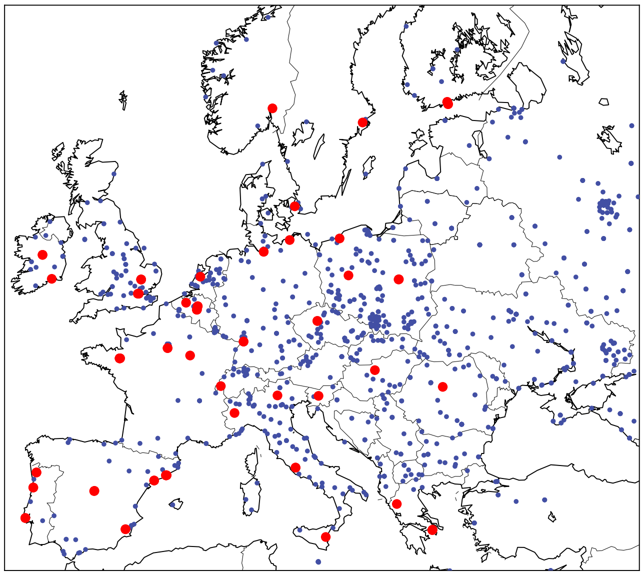

Figure 7: PlanetLab servers (red) and Speedtest.net servers (blue) used in the latency study.

used timing hooks injected into the Capture SDK to measure the time from starting a frame capture to an encoded frame ready for transport. We ran the Unigine Heaven graphics benchmark on the virtual machine while capturing the video stream using the SDK. The average delay for a single frame was 17 ms. As the purpose of this paper is to analyze the achievability of imperceptible latency in mobile cloud gaming in optimal conditions, we choose the 17 ms as a reference delay for the cloud gaming server.

3.4 Summary of different scenarios

The delay measurement results are summarized in Table 5. The results include all other delays of mobile cloud gaming except the wired segment of the network path. It is clear that the newer and more powerful Samsung Galaxy S7 is faster in all scenarios with an average of 25% or 37 ms improvement over all the test cases. This improvement cumulates mainly from faster touch interface, faster video decoding capabilities and a clearly faster gyroscope. In the optimal conditions using WiFi can save approximately 10 ms on the total delay as the access delay is significantly shorter. This improvement can be even greater depending on the network conditions.

The control method has a significant effect on the total delay as can be seen from the results. The most commonly used input method using the touch screen of the mobile device has quite a large impact on the total delay. There is however a substantial improvement in the newer S7 which receives a touch input in 27.5 ms on average to the application code compared to the 46 ms of the older S4. USB connected controller is the fastest method of receiving input commands into the application code adding only a couple of milliseconds to the delay. The gyroscope on the S4 is clearly not suitable for delay sensitive applications while the gyro on the S7 adds only 12 ms of delay to the pipeline. This shows that certain games with continuous movements such as car games could benefit from using the gyroscope of the mobile device instead of a virtual gamepad on the touch screen.

Table 5: Delay measurement results for all scenarios without wired segment of the network delay.

Mobile phone	Network type	Input type	Device delay	Server delay	Access delay	Total
S4	WiFi	Touch	110	17	1.6	128.6
		USB	70.4	17	1.6	89
		BLE	86.9	17	1.6	105.5
		Gyro	148.2	17	1.6	166.8
	LTE	Touch	110	17	12	139
		USB	70.4	17	12	99.4
		BLE	86.9	17	12	115.9
		Gyro	148.2	17	12	177.2
S7	WiFi	Touch	77.5	17	1.6	96.1
		USB	51.5	17	1.6	70.1
		BLE	72.5	17	1.6	91.1
		Gyro	62.7	17	1.6	81.3
	LTE	Touch	77.5	17	12	106.5
		USB	51.5	17	12	80.5
		BLE	72.5	17	12	101.5
		Gyro	62.7	17	12	91.7

Bluetooth controller on the S7 has approximately the same delay as the touch screen. However on the S4 the Bluetooth delay is actually 5 ms shorter than on the S7. This can be explained through the different Bluetooth configurations in the versions of the Android operating system. The newer Android 6.0 accepts slightly higher minimum intervals for updates on the BLE connection than the older Android 5.0.

The different delay scenarios are visualized in Figure 8 without the wired segment of the network path. The JND threshold for imperceivable delay for indirect touch (96 ms) is plotted as a dashed line to the figure. With the Samsung S4 phone the only scenario which leaves any room for the delay induced by the wired segment of the network path is the WiFi option with USB input. Same options are also best for the S7, however also the WiFi with Bluetooth and gyro also manage to fall under the threshold for perceivable delay. Using LTE, only the USB and gyro control options fall under the limit.

4. OPTIMIZING SERVER PLACEMENT: CASE EUROPE

In this section we measure the delay in the wired network path of the cloud gaming pipeline. This is the last piece missing from the total delay. We conduct a latency study in Europe and analyze how much the delay can be shortened by increasing the number of data center locations. The results are compared to the limits found for the different scenarios measured in Section 3. The found placements for the data center locations are also compared to existing and future Amazon EC2 data center locations.

4.1 Optimizing data center location count

The 79 Planetlab servers depicted in Figure 7 were chosen as the candidates for the locations of the data centers for the cloud gaming provider. The Eurostat statistics used also provide population information for each area with the speedtest.net servers for which we studied the network latency from the Planetlab server locations. This enabled us to have a database of geographical regions in Europe with latency data to the selected cloud gaming server locations with an estimate of the relative potential user data base in each region.

We utilized a greedy algorithm to rank the servers based on their coverage of potential users. The greedy algorithm

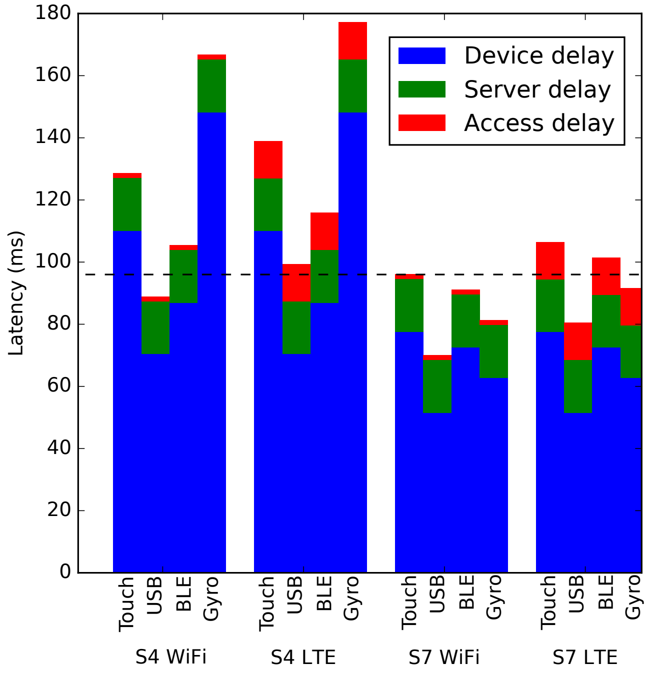

Figure 8: Response delay in different scenarios without the wired segment of the network path. The JND threshold for imperceivable delay (96 ms) is plotted with a dashed line.

has been previously proven to achieve near-optimal performance[29]. It approximates the global optimal solution by making the locally optimal decision in each stage. The greedy algorithm works as follows. In the first step, we choose a single server location which leads to the lowest average delay for all clients under the assumption that all the clients connect to that server. We then iteratively add a single server location in each step to the system which in conjunction with the previously chosen servers lead to the best average delay. We assume that each client connects to the closest server location. We add server locations into the system one by one until the latency benefit becomes negligible. We were also able to calculate the 4 first optimal data center locations with brute force and the results were identical with the greedy algorithm.

Figure 9 shows how the latencies reduce as more data center locations are added into the system. The results show that the latency difference between the 90th percentile from one data center location to ten locations is roughly 20 ms. After this each new data center location barely affects the latency distribution at all.

The results show that additional server locations above 10 don't shorten the average delay concieved by the service users significantly. The locations chosen by the greedy algorithm are shown in Figure 10. The number before the city name shows the priority of the location if less than 10 locations are to be chosen. As expected, central regions of Europe are more densily covered by chosen locations as the population density is larger in these regions. However as we did not impose capacity limits to the data centers, the first four locations are quite sparsely located to optimize the latency for all parts of the continent.

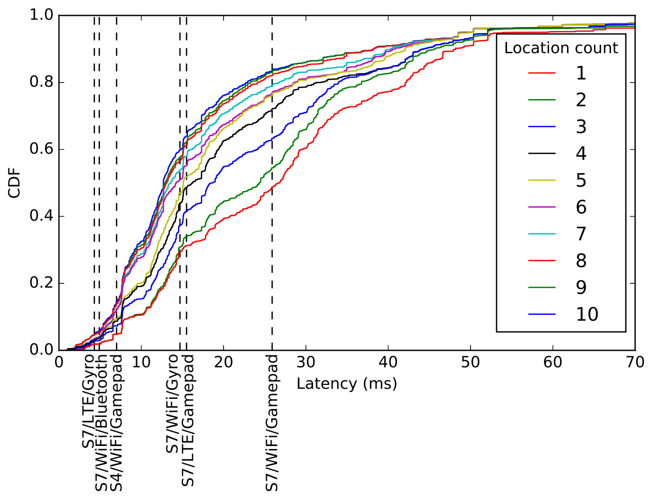

Figure 9: Latency distribution with different location counts

Using data from Section 3 and the 96 ms JND threshold, we can calculate what is the limit of perceivable delay for the wired network delay in different scenarios. The limits are drawn in Figure 9 with dashed lines. We notice that with the newer Samsung S7 phone using WiFi and USB input we can reach latencies below the limit for approximately half of the population using one data center location. The coverage can be raised to 60% with 3 data center locations and to 80% using 10 locations. With the same device and WiFi also the gyro input and USB input with LTE scenarios are somewhat feasible for a proportion of the population ranging from 30% with one location up to 60% with ten locations. In other scenarios it is practically impossible to achieve low enough latencies for any number of users.

Setting up a cloud infrastructure is possible only for the biggest companies in the entertainment business. Smaller companies most likely would at least in the beginning like to use existing cloud providers for their server locations. In Figure 11 we compare the latency distribution when using existing and future Amazon EC2 locations and the locations chosen with the greedy algorithm. At the time of writing the available locations for EC2 servers in Europe were Ireland and Frankfurt with London and Paris to be added soon. We compare both existing and future EC2 locations against the same number of locations placed by the greedy algorithm.

For the S7/WiFi/Gamepad scenario using locations calculated by the greedy algorithm, no additional users can achieve delays under the required limit compared to the EC2 locations. However the coverage can be raised from 50% to 70% by using four locations calculated by the greedy algorithm instead of the future Amazon EC2 locations. This shows that the current and future EC2 locations are possibly not completely optimized for latency. This is natural as there are multiple other requirements for a data center location and most of the latency-related aspects can be covered by using content delivery network (CDN) locations placed more on the network edge.

In our latency study so far, we have not assigned a maximum size for a data center location. This can lead to vastly different amount of users server by different data center locations. Next, we study how the latency distribution differs

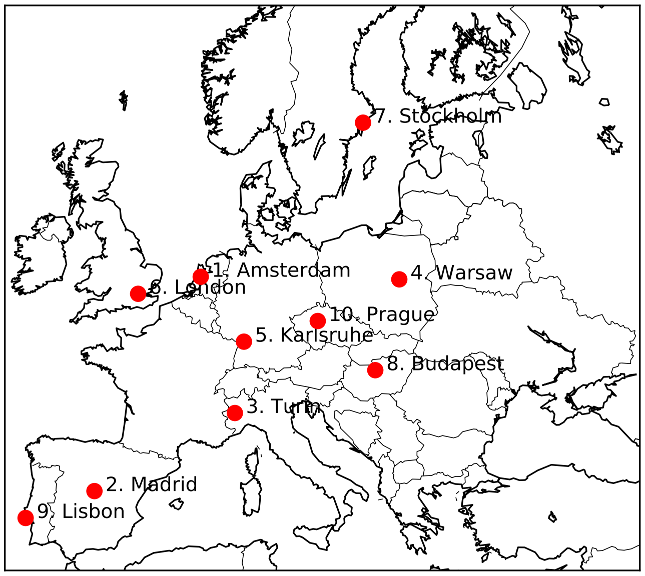

Figure 10: Server locations allocated by the greedy algorithm.

when the load on data centers locations is more evenly divided.

4.2 Load balancing and local optimization

The delay results presented in Figure 9 assumed all users would be served by their closest data center location. This is however not always possible if a certain location is under heavy load. To measure the effect of load balancing on the overall latency distribution, we perform a simulation of a cloud gaming system where the location sizes have been calculated so that under peak hours the users are evenly distributed across the data center locations. This means that in the peak hours a location with the lowest latency for a user might be full which leads to the user being assigned for the next location with longer latency until a free slot is available. We use the WoWAH dataset [23] to get realistic information on user churn in a game. The dataset includes 91065 players in total for which we assigned locations in Europe based on the population distribution for different areas.

The latency distribution for the load balanced simulation is compared with the non-balanced version in Figure 12. The median latency differs by under 5 ms from the non-balanced version. The latency range widens also a little on the balanced version, not however more than 5 ms.

The dataset used enables us to also calculate how much the server count can be lowered if local optimizations are performed using migrations inside a data center. The costs increase depending on the amount of servers the cloud gaming provider needs to set up. If the servers are rented, then the number of running instances also affects the overall cost. The g2.2xlarge instance used can handle 2-16 concurrent users at a time[26]. We used 4 as a realistic number of concurrent users for a high-end game as the instance has two GPU cores. However, as users leave the service, the game servers might have empty slots and the player distribution among the servers might become unnecessarily scattered. If local migration is possible, then the empty slots could be

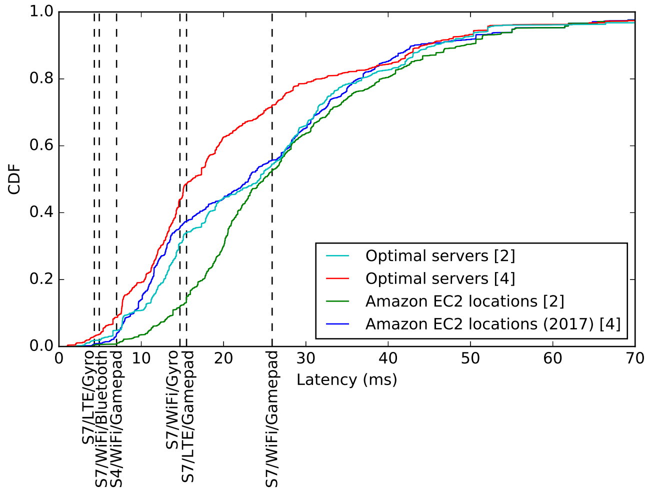

Figure 11: Latency distribution comparison between Amazon EC2 locations and locations given by the greedy algorithm.

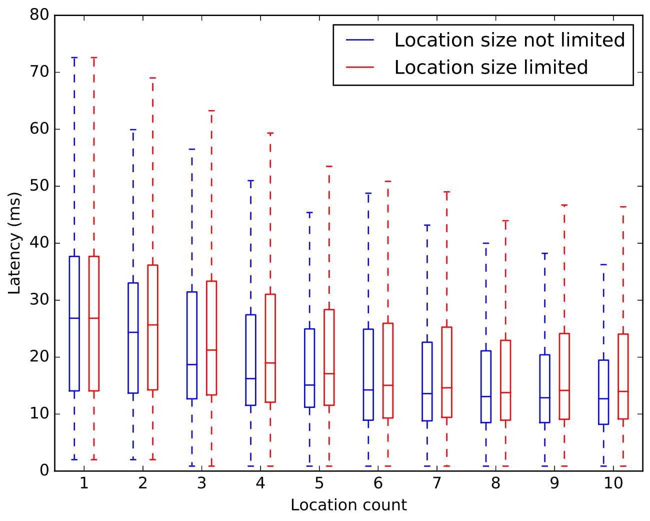

Figure 12: Effect of load balancing on latency distribution with different location counts.

filled on-the-fly and unnecessary servers shut down. We run a simulation using the dataset to calculate the possible cost savings if local migration is possible in the system. In the simulation, the servers were optimized each hour by migrating users between servers to minimize the total number of servers running. The simulation showed that the amount of servers running can be decreased by 13% by average if migrations are enabled. However the maximum number of servers required only decreased by 4%.

5. FUTURE PROJECTIONS

The results shown in Chapters 3 and 4 show that achieving imperceptible latency with current technology is difficult and requires optimal conditions with the correct type of control method. The base latencies of the mobile devices are still surprisingly high although definitive development from previous device generations has occurred. The touch screen and displaying a frame on the screen of the mobile device are the largest delay components that still need to be developed.

Previous study by Beyer et al. reveals how the touch screen latency has the tendency to decrease over device generations[3]. Frame display times are bound by the display refresh rate and by the use of double and triple buffering. Mobile device displays currently have a set refresh rate of 60 Hz which translates into a lowest possible delay of 16.7 ms between frames. In order to avoid visible tearing in the image, the operating system, including Android, uses double or sometimes triple buffering for holding previous rendered frames before they are actually displayed on the screen of the mobile device. This results in better visual quality with the expense of latency.

It is difficult for us to forecast whether the trend in reducing the touch screen latency continues in the future but some solutions to reduce the frame display latency are already available. One option is to increase the refresh rate of the device display. SoCs (system-on-a-chip) which support higher frequency (120 Hz) refresh rates have been already introduced[19]. They are not however present in current production devices.

Another option is to enable on operating system level the possibility for front buffer rendering. We measured a delay of 1.5 display periods or roughly 25 ms for display updates on current mobile phones showing the existence of double buffering. Our results suggest that front buffer rendering with a method called scanline racing could potentially drop this delay to 8 ms[2]. On the other hand, a display with 120 Hz refresh rate could reduce the latency of buffered updates by half compared to currently used displays. Cloud gaming clients also benefit from more powerful SoCs on mobile phones as the frame decoding time is shorter. However, the screen resolutions are also rising which could diminish the effect of more powerful SoCs.

Another source of delay that is likely to reduce in the future is the one caused by the access network. 5G mobile networks are aiming for shorter than 1ms latency. Such low latency is achievable through novel radio access technologies tapping the currently largely unused higher frequency spectrum, which allows for much wider channels and, consequently, much higher data rates as a result of which also the frame delivery delay reduces[30].

We plot the hypothetical latency results with front buffer rendering (FB), 120 Hz displays (Hz), and 1 ms access network delay (Ac) together with the current measured ones for the case of Samsung S7 device with LTE access in Figure 13. Just by employing front buffer rendering with a fast display reduces the latency by roughly 20 ms regardless of the control method. Looking back at Figure 11, such improvement would have a substantially bigger effect than doubling the number of server locations. In the case of clients using LTE and gamepad, for example, it would increase the fraction of clients that could be potentially served with imperceptible latency by 40-50 percentage points, depending on the server placements chosen, because the network latency budget increases from roughly 15 to 35 ms. Doubling the number of server locations increases the fraction of clients by 20-25 percentage points at most.

6. CONCLUSION

This paper studies the end-to-end latency in mobile cloud gaming. We set out to investigate whether it is possible

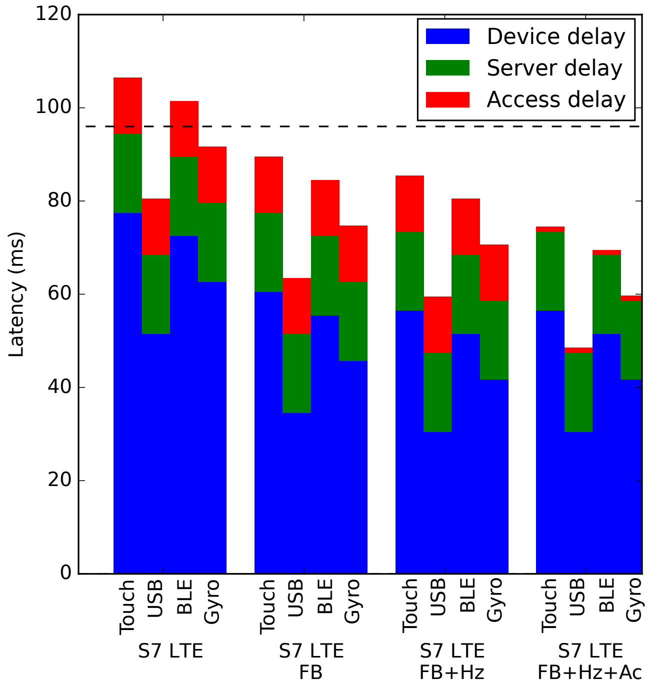

Figure 13: Response delay in different scenarios with future improvements without the wired segment of the network path. FB = Front buffer rendering. Hz = 120 Hz displays. Ac = 1 ms Access delay (5G)

to achieve latencies short enough for users not to perceive them while using the system. Our results suggest that it is possible but only in limited situations where, for instance, latest mobile device models and a separate control device are used because touch screen delay is too long. Replication of the service and server placement optimization are also necessary and they can have a major effect in providing short enough latencies. However, we also noticed that their impact is substantially smaller than the expected impact of near future developments on the mobile device side just because the mobile device imposed delays are clearly the biggest components in the end-to-end latency. In the future, it would be interesting to study in more detail the ability of users to perceive latency in mobile cloud gaming, extend the case study to other continents, and to investigate the effect of new VR and AR devices, such as Oculus Rift and Hololens, on the end-to-end latency.

7. ACKNOWLEDGMENTS

This work has been financially supported by the Academy of Finland (grant numbers 278207 and 297892), Tekes - the Finnish Funding Agency for Innovation, and the Nokia Center for Advanced Research.

8. REFERENCES

[1] 3rd Generation Partnership Project. *TS 36.214 LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer; Measurements*, 2014. Release 12.

[2] Anonymous. Anonymized for double-blind review.

[3] J. Beyer, R. Varbelow, J.-N. Antons, and S. Zander. A method for feedback delay measurement using a low-cost arduino microcontroller. In *Proceedings of the 7th International Workshop on Quality of Multimedia Experience (QoMEX), IEEE*, 2015.

[4] K. Boos, D. Chu, and E. Cuervo. Flashback: Immersive virtual reality on mobile devices via rendering memoization. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 291–304, New York, NY, USA, 2016. ACM.

[5] E. Cattan, A. Rochet-Capellan, and F. Bérard. A predictive approach for an end-to-end touch-latency measurement. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ITS '15, pages 215–218, New York, NY, USA, 2015. ACM.

[6] C.-M. Chang, C.-H. Hsu, C.-F. Hsu, and K.-T. Chen. Performance measurements of virtual reality systems: Quantifying the timing and positioning accuracy. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 655–659. ACM, 2016.

[7] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu. On the quality of service of cloud gaming systems. *Trans. Multi.*, 16(2):480–495, Feb. 2014.

[8] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1269–1272. ACM, 2011.

[9] S. Choy, B. Wong, G. Simon, and C. Rosenberg. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia Systems*, 20(5):503–519, 2014.

[10] J. Deber, B. Araujo, R. Jota, C. Forlines, D. Leigh, S. Sanders, and D. Wigdor. Hammer time!: A low-cost, high precision, high accuracy tool to measure the latency of touchscreen devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2857–2868. ACM, 2016.

[11] J. Deber, R. Jota, C. Forlines, and D. Wigdor. How much faster is fast enough?: User perception of latency & latency improvements in direct and indirect touch. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1827–1836. ACM, 2015.

[12] Eurostat. NUTS - Nomenclature of territorial units for statistics. Overview. http://ec.europa.eu/eurostat/web/nuts/overview.

[13] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. Qoe-aware virtual machine placement for cloud games. In *Network and Systems Support for Games (NetGames), 2013 12th Annual Workshop on*, pages 1–2. IEEE, 2013.

[14] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. Placing virtual machines to optimize cloud gaming experience. *Cloud Computing, IEEE Transactions on*, 3(1):42–53, Jan 2015.

[15] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen. Gaminganywhere: An open cloud gaming system. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 36–47, 2013.

[16] Z. Ivkovic, I. Stavness, C. Gutwin, and S. Sutcliffe. Quantifying and mitigating the negative effects of local latencies on aiming in 3d shooter games. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 135–144, New York, NY, USA, 2015. ACM.

[17] P. Jain, J. Manweiler, and R. Roy Choudhury. Overlay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, pages 331–344, New York, NY, USA, 2015. ACM.

[18] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hossfeld. An evaluation of qoe in cloud gaming based on subjective tests. In *Proceedings of the 5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS),*, pages 330–335. IEEE, 2011.

[19] B. C. Joshua Ho. MediaTek Demonstrates 120 Hz Mobile Display. http://www.anandtech.com/show/8852/mediatek-demonstrates-120-hz-mobile-display.

[20] M. Koudritsky. Github. Walt Latency Timer. https://github.com/google/walt.

[21] B. Lee and A. Oulasvirta. Modelling error rates in temporal pointing. In *Proc. of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1857–1868. ACM, 2016.

[22] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, pages 151–165, New York, NY, USA, 2015. ACM.

[23] Y.-T. Lee, K.-T. Chen, Y.-M. Cheng, and C.-L. Lei. World of warcraft avatar history dataset. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 123–128. ACM, 2011.

[24] Y.-T. Lee, K.-T. Chen, H.-I. Su, and C.-L. Lei. Are all games equally cloud-gaming-friendly? an electromyographic approach. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, pages 1–6. IEEE, 2012.

[25] NVIDIA. NVIDIA Capture SDK. https://developer.nvidia.com/capture-sdk.

[26] Nvidia. Nvidia GRID GPU Specs and features. http://www.nvidia.co.uk/object/cloud-gaming-gpu-boards-uk.html.

[27] L. Pantel and L. C. Wolf. On the impact of delay on real-time multiplayer games. In *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '02, pages 23–29. ACM, 2002.

[28] PlanetLab. PlanetLab Europe. https://www.planet-lab.eu/.

[29] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1587–1596. IEEE, 2001.

[30] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz,

M. Samimi, and F. Gutierrez. Millimeter wave mobile communications for 5g cellular: It will work! *Access, IEEE*, 1:335–349, 2013.

[31] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[32] R. Shea, J. Liu, E.-H. Ngai, and Y. Cui. Cloud gaming: architecture and performance. *IEEE Network*, 27(4), 2013.

[33] O. Soliman, A. Rezgui, H. Soliman, and N. Manea. *Mobile Cloud Gaming: Issues and Challenges*, pages 121–128. Mobile Web Information Systems. Springer, 2013.

[34] Speedtest. Speedtest.net. http://www.speedtest.net/.

[35] D. Wu, Z. Xue, and J. He. icloudaccess: Cost-effective streaming of video games from the cloud with low latency. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(8):1405–1416, Aug 2014.

[36] Q. Zhang, Q. Zhu, M. Zhani, R. Boutaba, and J. Hellerstein. Dynamic service placement in geographically distributed clouds. *Selected Areas in Communications, IEEE Journal on*, 31(12):762–772, December 2013.

[37] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, and J. L. Hellerstein. Dynamic service placement in geographically distributed clouds. *IEEE Journal on Selected Areas in Communications*, 31(12):762–772, 2013.