# Lag Compensation for First-Person Shooter Games in Cloud Gaming

Zhi Li[1], Hugh Melvin[1(✉)], Rasa Bruzgiene[2], Peter Pocta[3], Lea Skorin-Kapov[4], and Andrej Zgank[5]

[1] National University of Ireland, Galway, Ireland
hugh.melvin@nuigalway.ie
[2] Kaunas University of Technology, Kaunas, Lithuania
[3] University of Zilina, Zilina, Slovakia
[4] University of Zagreb, Zagreb, Croatia
[5] University of Maribor, Maribor, Slovenia

**Abstract.** Cloud gaming is an emerging technology that combines cloud computing with computer games. Compared to traditional gaming, its core advantages include ease of development/deployment for developers, and lower technology costs for users given the potential to play on thin client devices. In this chapter, we firstly describe the approach, and then focus on the impact of latency, known as lag, on Quality of Experience, for so-called First Person Shooter games. We outline our approach to lag compensation whereby we equalize within reason the up and downlink delays in real-time for all players. We describe the testbed in detail, the open source Gaming Anywhere platform, the use of NTP to synchronise time, the network emulator and the role of the centralized log server. We then present results that firstly validate the mechanism and also use small scale and preliminary subjective tests to assess and prove its performance. We conclude the chapter by outlining ongoing and future work.

**Keywords:** Cloud gaming · Quality of experience · Network delay
Lag compensation

## 1 Introduction

The gaming industry plays an important role in the entertainment and software industries. According to "Video Game Revenue Forecast: 2017-22", it is expected that the global market of video games will grow up to $174 billion by 2022 [1].

Traditionally, computer games are downloaded from the Internet and installed on a PC or other end user device allowing players to run the corresponding game. With game sizes running into multiple gigabytes, the installation process may take the order of hours, with perhaps additional time required to install patches of new game versions. Furthermore, when players wish to play newly released games, they may require a higher specified hardware configuration to enable all the visual effects, and so they have to upgrade their computers to meet the particular specification. Both of these factors can result in frustration and may result in gamers give up the game [2].


I. Ganchev et al. (Eds.): Autonomous Control for a Reliable Internet of Services, LNCS 10768, pp. 104–127, 2018.
https://doi.org/10.1007/978-3-319-90415-3_5

Unlike conventional computer games, cloud gaming has a different paradigm. The 'heavy lifting' of game processing is done by servers in the cloud [3]. Game actions are captured by game clients and sent to cloud server(s). The resulting game scenes are rendered by the cloud servers and the audio and video frames are streamed back to clients over the broadband network. Gamers thus interact and control games through thin clients, the thin client being a lightweight process (often a browser) which interacts with the remote server [2]. Figure 1 shows the relationship between the server and client in a cloud gaming service, with gamer actions captured and sent to the cloud-based gaming provider which then streams a video back to the client. For this reason, cloud gaming allows gamers to play games with simple devices (also referred to as thin client devices) without having to install the games or to continuously upgrade computer hardware.

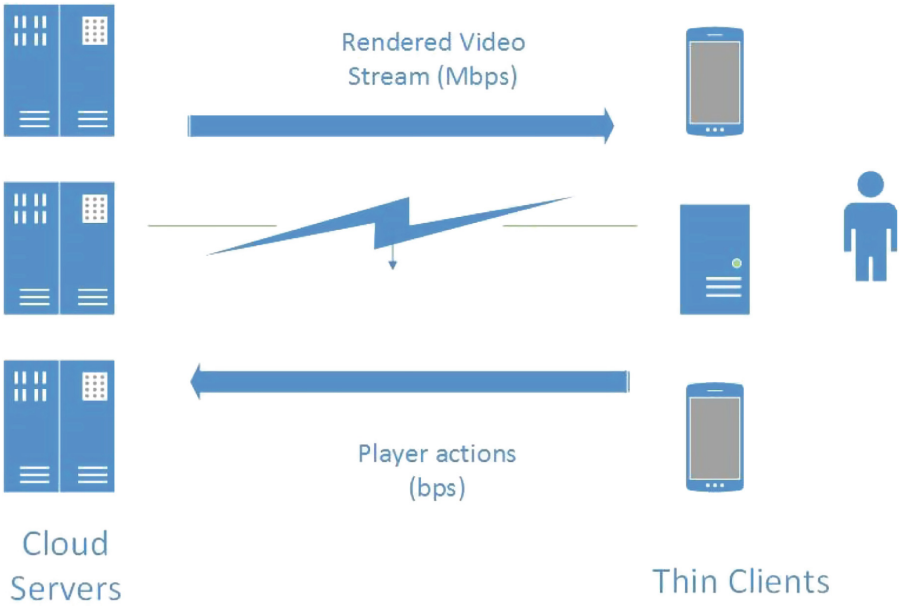


**Fig. 1.** Cloud gaming service

For these reasons, game developers and users/gamers are paying more attention to cloud gaming systems [3]. From the developer perspective, the benefits include the potential of reaching out to more gamers, easier testing of ideas to improve cloud gaming systems, avoiding piracy due to the fact that games are not being downloaded to client devices [4].

From the gamer perspective, the benefits include access to games anytime (on demand), without the need to download games, reduced costs due to the fact that the computer hardware does not need to be upgraded frequently, and ability to play games on different platforms, such as PC, smartphone, tablet and so on.

Although cloud games open up a new direction for the video games industry, it is not without its challenges [2]. According to previous research, not only the bandwidth but also the CPU has a significant influence on cloud gamers' Quality of Experience (QoE) [5]. Ideally, gamers would like to play games with both high quality videos, and where games are delay sensitive, low latency. Latency, also known as lag in gaming, is especially important in the First Person Shooter (FPS) game genre. However - high quality videos, for example, 720p/1080p at 50 fps, can make cloud gaming systems vulnerable to a high network latency [3] as much more network capacity is needed than in the case of conventional video games (e.g., 5000 kb/s vs. 50 kb/s) [3]. To meet the needs of cloud gamers, network service providers thus have to take network latency, efficiency, high video quality, and error resiliency into consideration [6]. These factors represent significant challenges to the roll out of large scale cloud gaming services. Without adequate infrastructure that meets the specific needs of cloud gaming, the potential and benefits will not be realised.

In this chapter, we focus on the impact of lag on QoE for so-called FPS games. In multi-player scenarios, different lag values experienced among players can lead to unfair game play and frustration among players [7]. In [8], the authors report findings that state that different QoS leads to unfairness or imbalanced games when there are no mechanisms for mitigating the QoS differences. Previous research has analysed the potential of achieving fairness in multi-player networked games through automated latency balancing [9]. We further tackle these challenges in the context of cloud gaming. We describe in detail our approach to lag compensation whereby we equalise within reason the up and downlink delays in real-time for all players, aiming to achieve fairness among players and consequently improve QoE.

At present, several cloud service providers have developed cloud gaming platforms, most of which are closed source (e.g., Sony PlayStation Now, NVIDIA GeForce Now). Therefore, game developers cannot test and fine-tune their games on them [3] and they have to do the tests and fine-tuning of the games on emulators. This fact increases the difficulty of improving cloud gaming systems to better reflect gamers' needs and expectations.

In response to this, Gaming Anywhere (GA) was designed and developed. It is the first open-source cloud gaming platform that allows researchers to quickly explore their ideas. More importantly, GA has greatly promoted the evolution of cloud gaming within the video game industry [3].

The remainder of this chapter is structured as follows. Section 2 provides a literature survey on gaming QoE. It firstly deals with the impact of latency and packet loss on game QoE, before focusing specifically on latency. It then examines the impact of delay for different game genres. Section 3 introduces the concept of lag compensation and outlines our two research objectives. Section 4 outlines in detail how lag equalization was implemented on the Gaming Anywhere platform in order to address both research objectives. Section 5 presents results that firstly validate the lag compensation mechanism and then outlines the results of preliminary subjective tests. Section 6 concludes the paper by outlining ongoing and future work.

## 2 Literature Review

A key research challenge has been to determine the impact of a wide range of influence factors on gaming QoE, including a wide range of human, system, and context factors [10–12]. Focusing on system influence factors, and as outlined in the previous section, cloud gaming demands a high level of network Quality of Service (QoS) to deliver acceptable user perceived quality (QoE) to players. Key QoS-related factors include packet loss and delay, with their impact on QoE differing for different types (genres) of games.

In this section, we review relevant literature that outlines the impact that both roundtrip delay (latency), also known as Response Delay (RD) or lag, and packet loss have on the end user experience in general, and also for different game genres. As the main focus of this chapter is on lag compensation, we focus more on lag in the literature review.

Given the inherently interactive nature of gaming in general, a key challenge is meeting delay requirements. This involves the delivery of both user control inputs (mouse, keyboard strokes) to the game server and uninterrupted presentation of continuous game content to players, transmitted in the form of a video stream. For this reason, conventional methods for diminishing the effects of poor network conditions and consequent jitter on streaming media, such as buffering data for display, are not readily applicable in the context of cloud gaming. Moreover, lag compensation techniques applicable in "traditional" gaming, such as client-side prediction [13], are not applicable in the context of cloud gaming, where the client is simply decoding and portraying the stream received from the server. Consequently, numerous studies have addressed the impact of latency due to heterogeneous and variable network conditions on the end user QoE of cloud gaming.

As reported in [5, 14–16], packet loss and delay have a significant impact on cloud gaming QoE. Basically, network congestion results in network delay jitter and when queue size is exceeded, packet loss occurs. Delay and jitter impact both on uplink time between the player sending input events to the server, and downlink transmission of game scenes that are eventually displayed on the screen. Moreover, as it has been shown in [17], a high network delay disrupts an interaction between server and players and negatively influence players' QoE.

It is important to note that not all cloud games are equally sensitive to latency, as is of course also the case for "traditional" networked games [18]. For Real-time Strategy (RTS) games, the process of constructing buildings or moving troops towards a battlefield is unaffected by latency as high as 1000 ms [15]. However, First Person Shooter (FPS) games, where users are shooting at a moving target tend to be more sensitive to latency with delays of over 100 ms seen as unacceptable [19]. Moreover, the effects of latency are based on two action properties: precision and deadline. Precision refers to the accuracy of actions, whereas deadline refers to the timeliness of events. Games with higher precision and tight deadline are more sensitive to latency. For this reason, FPS players always emphasis precision and deadline [20].

As it has been reported above, latency plays a very important role when it comes to cloud gaming. Despite this fact, there is no work, to the best of our knowledge, dealing

specifically with the impact of delay compensation on QoE in the context of cloud gaming. Therefore, we have decided to focus on this issue in this chapter. More specifically, we showcase lag compensation impact on QoE in a case study involving an FPS cloud game.

## 3 Lag Compensation

Figure 2 shows the relationship between the server and the client for cloud gaming. The client sends control events to the server over the network, then the server samples/executes the input commands and delivers a stream (Audio/Video) back to the client. Finally, the client receives and decodes the stream to be portrayed on the screen. This round-trip delay is also known as response delay.

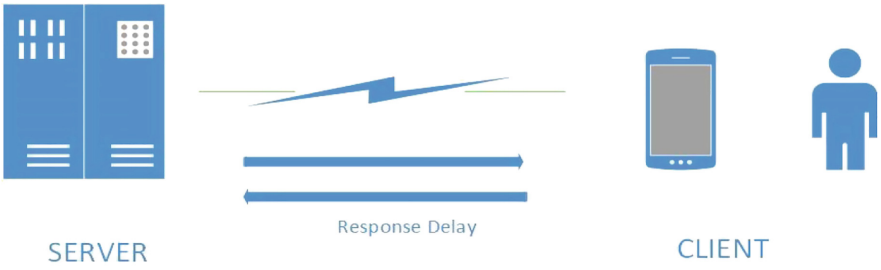


**Fig. 2.** The relationship between the server and the client

Basically, lag compensation is a technique that attempts to equalise lag for all players in a cloud gaming scenario. For example, in Fig. 3, there are two players (P1 and P2) that are playing an FPS cloud game.

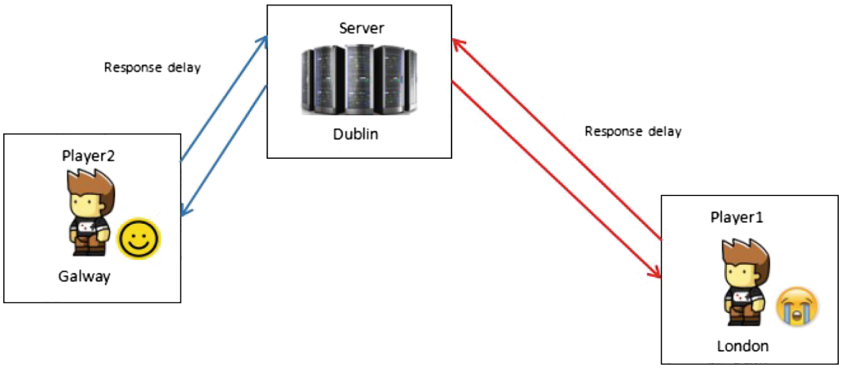


**Fig. 3.** Two players with different RD

The game server is located in Dublin, Ireland, P1 plays in London with an average lag of say 100 ms (equal delay in both directions) while P2 is in Galway, Ireland with an average lag of say 10 ms, again with equal delay in both directions. Since P1 has a longer RD than P2, P1 will have a relatively bad game experience as P2 has an inherent advantage.

To analyse and visualise this lag difference further considers Fig. 4, where the player in Galway (P2) has moved from Position A to Position B.

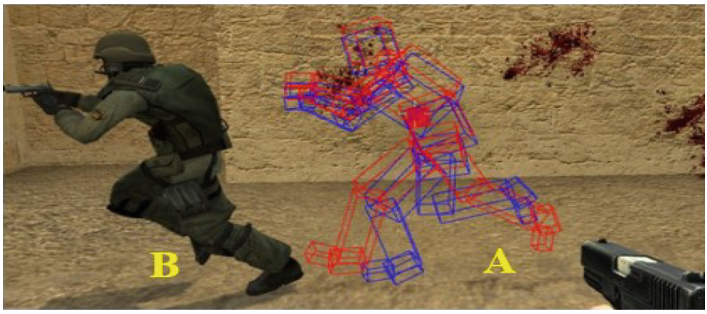


**Fig. 4.** Example of Lag in an FPS game (taken from https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking) (Color figure online)

The red hitbox shows the A position where P2 was prior to moving. However, due to longer RD for P1, his view of the game still shows P2 at position A. When P1 executes a shoot action, the gameplay information is sent to the server. When this command arrives at the server, the target P2 has already moved to position B. As a result, P1 misses P2 even though P1 correctly aimed at the opponent in his view of the game. To eliminate this issue, lag compensation is needed on server side, such that an artificial delay is added to P2 so that both P1 and P2 experience the same lag on both up and downlink traffic and the game thus becomes fairer. Figure 5 shows the game after lag compensation.

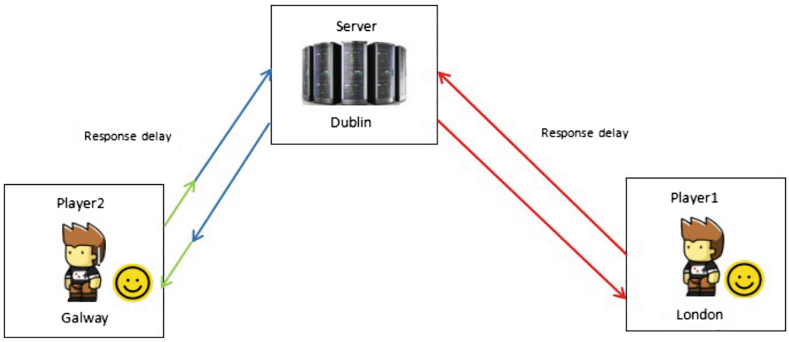


**Fig. 5.** Game after lag compensation

As shown, equal delays are added to both the uplink (client to server or c2s) and downlink (server to client or s2c) traffic. The assumption of equal delays in both directions is in reality rarely true, largely due to asymmetry in traffic flows.

### 3.1 Research Objectives

Having given a brief overview of literature relating to the QoE of gaming, and introduced the proposed lag equalization concept, we define two key research objectives that are the focus for the remainder of the chapter:

1. How feasible is it to implement a real-time lag compensation strategy for cloud gaming?
2. Will the lag compensation approach result in improved QoE for FPS gamers?

## 4 Implementation

In this section, we describe the implementation details related to the testbed used to address both research objectives. We firstly describe the cloud gaming platform Gaming Anywhere that was used, followed by the game that was chosen for the platform. Other testbed requirements and tools such as time synchronization and network emulator are also briefly described.

### 4.1 Infrastructure

As previously mentioned, GA is designed to better bridge the computer game industry and the research community. The most attractive feature is its openness, with GA being the first open-source cloud gaming platform. Unlike other existing systems, GA allows developers and researchers to explore their ideas on a real testbed and extend current system. As defined previously, the response delay RD is the time between GA client sending input events to GA server and the responding game scenes being displayed on the player's screen. Basically, the RD is composed of three components [21]:

- Processing delay (PD): it represents the time between when the server receives the control events and sends the encoded frames to the client.
- Playout delay (OD): it is the time to decode and render the decoded frames on the screen on the client side.
- Network delay (ND): it is the time required for a round trip data exchange between the client and the server.

With the platform chosen, the next step was to choose a suitable game with which to test/validate our approach. As outlined in the Literature Review, an FPS game was most suited as these have the tightest lag constraints, i.e., are more sensitive to lag than other game genres. For this reason, the game ***Assault Cube*** was chosen.

### 4.2 Game Setup

To test the basic operation of the approach, a two-player set up was chosen. To ensure that players are in the same game but have different game views, two instances of the GA server are needed, one per player. This means that each player (client) has its own server. One of these servers becomes the Master game server and the other a slave server. For this reason, 4 computers were needed to setup the experimental environment. For a more scalable implementation, a VM approach is required to run all GA servers as described later. Since all machines are in the same university lab and each of them connects to the network via IEEE 802.3u Fast Ethernet 100 Mbps switched network, network delays are minimal. Figure 6 shows the experimental environment.

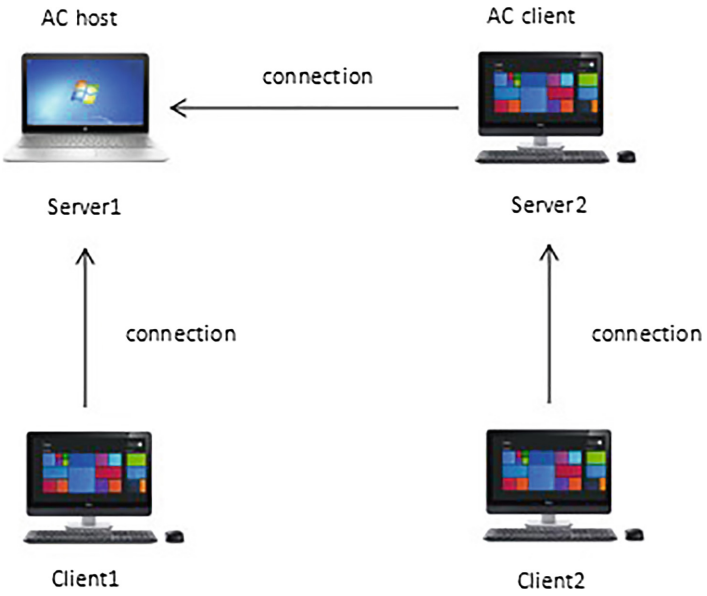


**Fig. 6.** The experimental environment

### 4.3 Assault Cube Configuration

Assault Cube (AC) is an FPS game which is based on the CUBE engine and available for free on Windows, Linux and OS X. It supports single player and multi-player game mode. AC is launched in Server1 first, and this becomes the master/host for a multi-player game scenario. AC is then launched in Server2, by selecting multi-player mode and joining the game created by Server1 as a slave. Figure 6 illustrates the connection between the two servers and also between the two GA clients/players and their respective GA servers. As shown in Figs. 1 and 2, the actual data flow is bidirectional. Once set up, each GA client sends its game actions to its server and receives video feed from its own server.

### 4.4 Time Synchronization

Network Time Protocol (NTP) is designed to synchronise system time of computers across IP networks to Universal Coordinated Time (UTC), achieving millisecond (ms) level synch or better across well provisioned wired LANs and single ms level synch across well configured WANs. In this experiment, NTP plays a key role in synchronising time across the GA servers and GA clients. This then facilitates accurate delay measurements as outlined in the next section. In our university LAN, the server to client (s2c) delay and client to server (c2s) are minimal with RTT measured via the ping utility of the order of a few ms or less. Due to NTP tolerances when running on Windows platform and resulting clock offsets, the s2c and c2s delays occasionally are determined as less than 0 or greater than the RTT.

For our testbed, Server1 was set as the reference clock (NTP server mode) with the other GA server and both GA clients setup as NTP clients. With this configuration, shown in Fig. 7, time offsets of around 1 ms were typical with occasional fluctuations. Although the synchronisation performance of NTP on Windows platform is not as good as on Unix type platforms, the levels of synchronization achieved (1–2 ms) are sufficient for the case where we are looking at network delays of 100 ms and more.

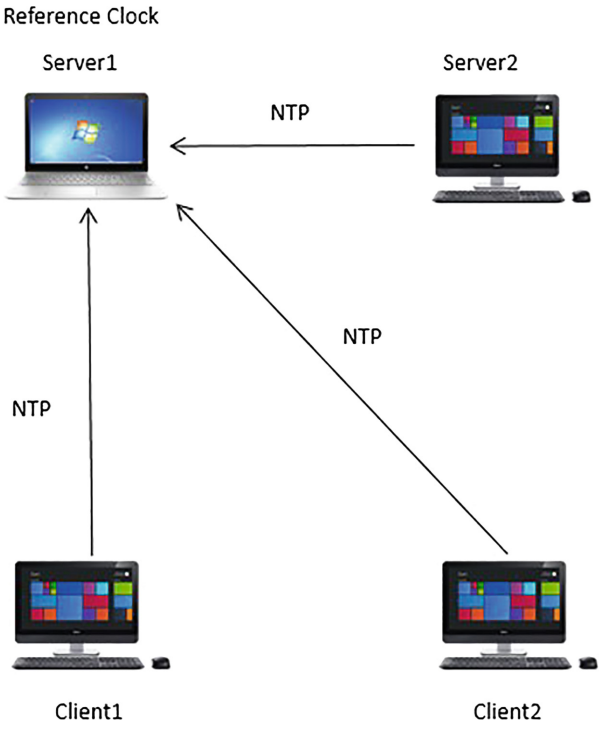


**Fig. 7.** NTP setup

### 4.5 Each Way Delay Measurement

The GA platform utilizes the Live555 Realtime Transport Protocol RTP library to transport audio/video from server to client. RTP and its companion control protocol RTCP are very widely used in VoIP conferencing software such as WebRTC and Facebook, WhatsApp voice clients. For our purposes, the RTCP library source code was modified to enable calculation of each way delay. By default, RTCP traffic, which runs in parallel to the media RTP flows for both audio and video, enable the calculation of Round Trip Delay, i.e., RTT minus residency time on remote host (described above as Network Delay in [21]). By adding code to also return the local timestamp when the RTCP receiver report packet is sent back, this facilitates the calculation of both upstream and downstream delay for each stream, once NTP is correctly implemented.

### 4.6 Network Emulator

In a real Cloud Gaming environment, the players are often at different geographic locations with different network latency, resulting in an unfair game for the player with longer RD. In order to emulate this scenario and thus test the implementation of lag equalization, Network Emulator for Windows Toolkit (NEWT) (available from https://blogs.technet.microsoft.com/juanand/2010/03/05/standalone-network-emulator-tool/) was used on the server side to emulate different network environments for both uplink (c2s) and downlink (s2c) traffic for each player.

### 4.7 Centralized Log Server

Since multiple GA server instances are required – one per player, a centralized log server is required to collect delay data in real-time from each GA server, perform QoE analysis, and then transmit required up and downlink lag compensation delays back to each corresponding GA server. The data sent from each server to the centralized log server includes synchronization source SSRC, server to client (s2c) delay, client to server (c2s) delay, IP address, and port number. Figure 8 illustrates the data generated and gathered by each GA server into a char array, and sent to a centralized log server in real-time.


**Fig. 8.** Data structure in GA server

Figure 9 illustrates the centralized log server within the whole architecture. Once it receives data, it will compare s2c delay and c2s delay between different GA servers, and determine what compensating delay to add on upstream (player control actions) and downstream (video/audio) in real-time for each player.

UDP sockets are used instead of TCP sockets to minimize delay and maximize responsiveness of system – it does not need any connection setup such as

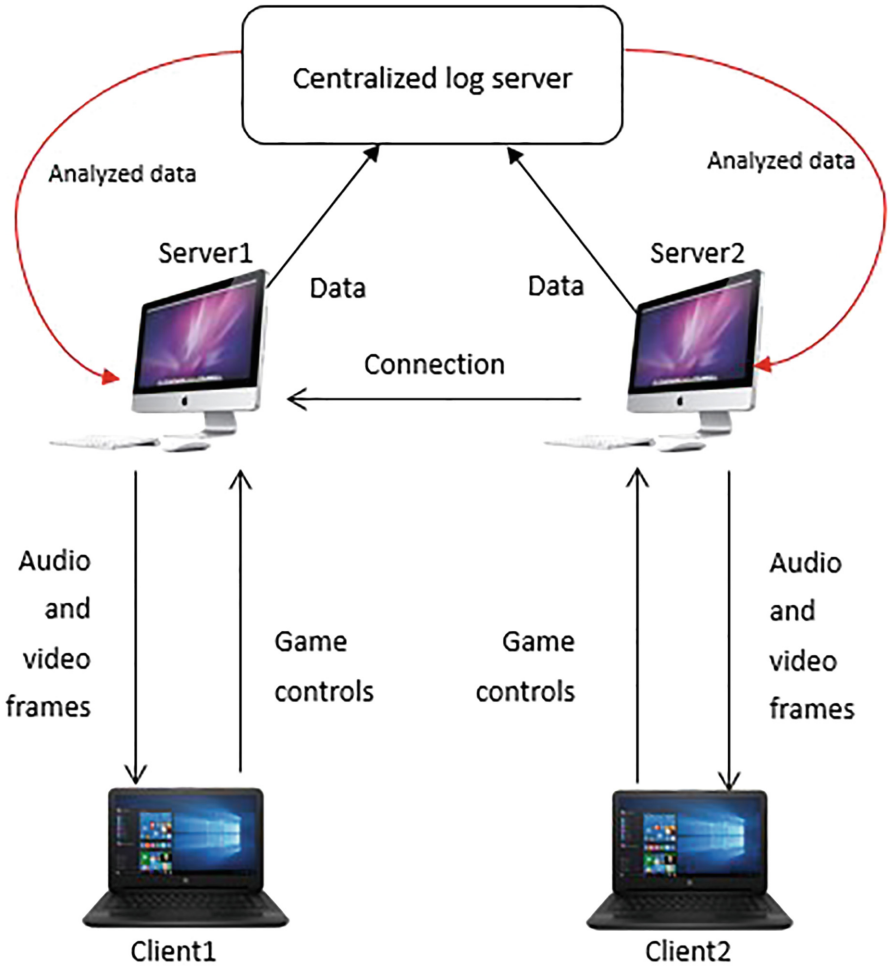


**Fig. 9.** Centralized log server

three-way-handshake, and ignores lost packets which would otherwise incur more delay when sending data. UDP sockets are thus an appealing choice for non-critical delay-sensitive applications.

Since data contains characters and integers, we use an object array to store data together. Figure 10 illustrates the data structure in centralized log server. Once the centralized log server receives data from more than 1 player, it starts to do the analysis and generates the table of data above. Firstly, from incoming data packets the centralized log server determines the player with the largest s2c delay for audio and video respectively. Based on upstream delays for RTCP RR traffic, it calculates an average (of video/audio) c2s delay also. As outlined earlier, the threshold of roundtrip delay for FPS games is 100 ms. For this reason, we implement a threshold such that whenever the round-trip delay is greater than 100 ms, the data for that player is not considered as

| | SSRC | s2c delay | Added down stream delay | c2s delay | Avg c2s delay | Added up stream delay | IP | Down stream port | Up stream port |
|---|---|---|---|---|---|---|---|---|---|
| Data[0] | .. | .. | .. | .. | .. | .. | .. | .. | .. |
| Data[1] | .. | .. | .. | .. | .. | .. | .. | .. | .. |
| Data[2] | .. | .. | .. | .. | .. | .. | .. | .. | .. |

**Fig. 10.** Data structure in centralized log server

it makes no sense to penalise every player with more than 100 ms round-trip time after lag compensation.

As shown in Fig. 11, Client1 and Client2 have 30 ms and 50 ms round-trip time respectively while client3 has 120 ms. Since the threshold is 100 ms, the centralized log server will just do comparison of data for Client 1 and 2.

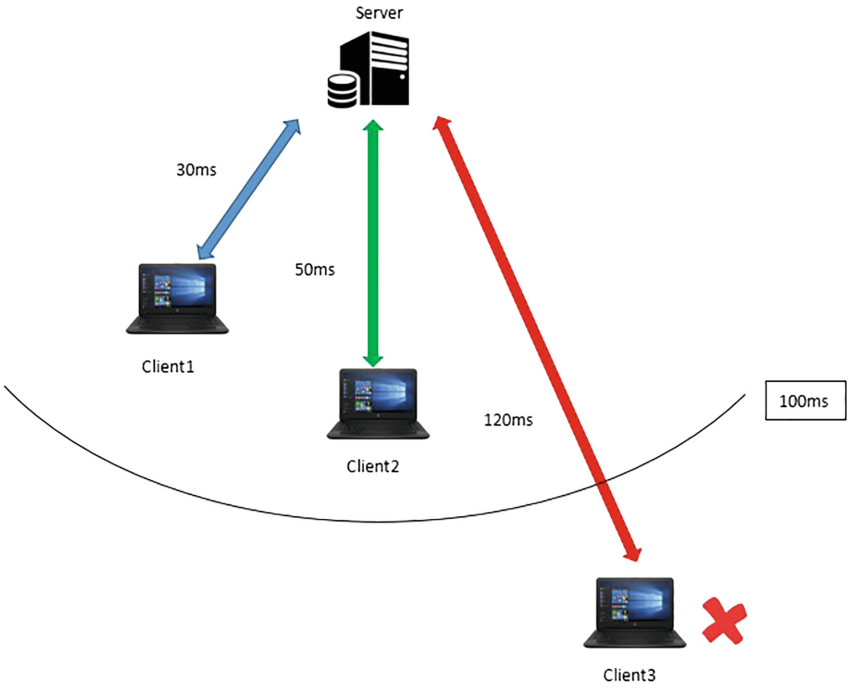


**Fig. 11.** Threshold network latency for FPS games

Once determined, the centralized log server sends recommended compensation delay data back to each GA server based on IP address. The respective GA servers then introduce these additional delays on upstream control traffic (player actions) and downstream data (audio/video).

Figure 12 shows the full structure of testbed in this experiment and also outlines the data flows as well as role of network emulator and lag compensation.

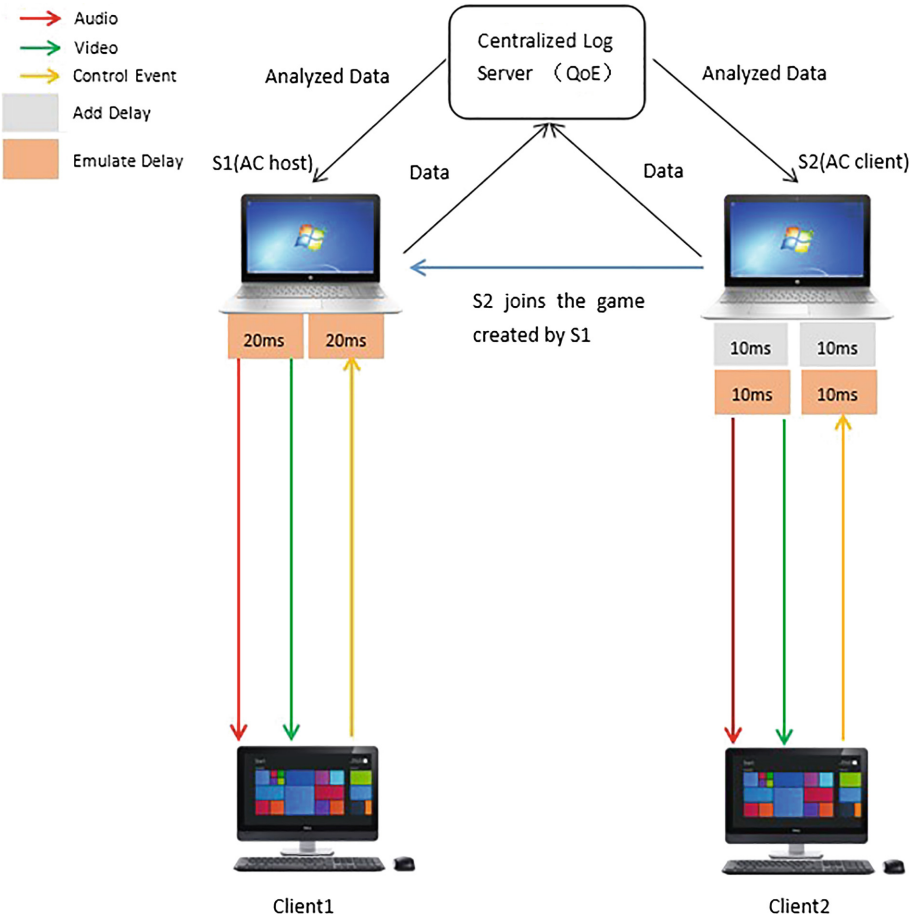


**Fig. 12.** Full system architecture and data flows

As shown above, we introduce emulated delays of 20 (10/10) and 40 (20/20) ms to Client1 and Client2 respectively. This results in lag compensation of 10 ms on both up and down link traffic for Client2 – shown in grey above so that c2s and s2c are equal for both clients. Note that the delays shown here are symmetric, which is rarely the case in reality. Further details on setting up the testbed and background work can be found in [22].

## 5 Results

In this section, we outline results that address both research objectives from Sect. 3.1. We firstly present a range of results for the 2-player scenario in order to validate the performance of the lag compensation mechanism. We present baseline delays with no emulated delays to get a sense for delays as measured across the university LAN network and the possible need for lag compensation to cope with inherent delay asymmetries. We then outline a range of tests whereby we introduce both up and downlink delays for different players using the network emulator and see how the lag compensation mechanism performs. Moving to objective 2, we then present results of preliminary subjective tests that exposed players to a range of delays with and without lag compensation and captured the resulting QoE scores. The section concludes with a discussion of ongoing and future work.

### 5.1 Baseline Results

Figure 13 shows the analysed data processed by the log server from the two servers/players deployed across the LAN network environment with no artificial delays added by the network emulator. Player 1 data is shown in the first 2 data rows, Data 1 for video and Data 2 for audio. Player 2 data is shown in the next 2 rows, Data 3 for video and Data 4 for audio. As outlined above, the columns (left to right) show SSRC (v for video and a for audio), s2c delay, delay added to downstream (s2c) traffic, c2s delay, average c2s delay (of video and audio stream), delay added to upstream traffic (c2s), IP address and 2 ports. The average c2s is calculated (from audio and video RTCP traffic) and used to delay the uplink control traffic.

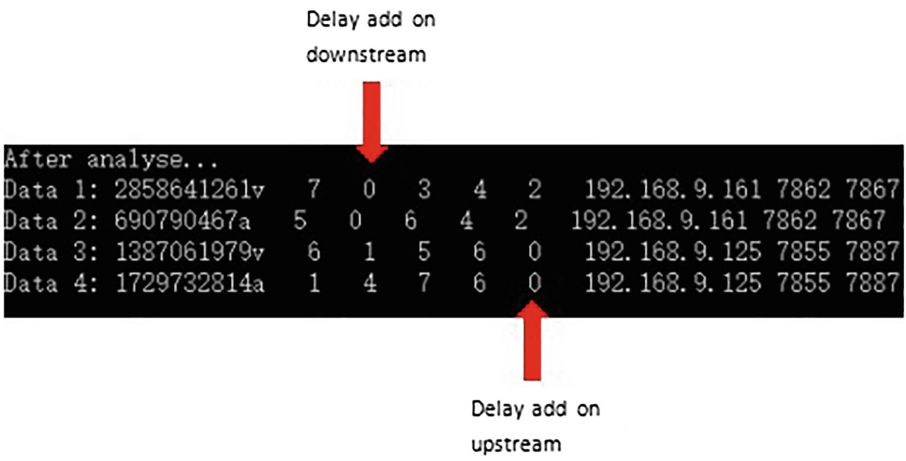


**Fig. 13.** Analysed data with no emulated delays

The output shows how the lag compensating mechanism is achieving the goal of equalising the delays for both players. For example, Player 1 measured s2c delays are 7/5 ms for v/a respectively whereas Player 2 has 6/1 ms for v/a respectively. Therefore, in order to equalise this, compensating delays for Player 2 s2c of 1/4 ms are added for v/a so that totals for Player 2 are 7 (1 + 6) and 5 (4 +1) i.e. same as Player 1.

Moving to c2s delays, Player 1 has measured c2s delays of 4/4 ms for v/a respectively whereas Player 2 has 6/6 ms for v/a respectively. Therefore, in order to equalise this, Player 1 c2s added delays are 2 for v/a so that total for Player 1 is 6 (4 + 2) i.e. same as Player 2.

### 5.2 Emulated Delays

In order to fully test the lag compensation mechanism, we then added delays using the NEWT emulator and monitored both how quickly these delays were picked up by the mechanism and then how compensating delays were added to equalise delays for both players. We firstly added 10 ms to both up and downlink delay for Player1/Client1 and 20 ms delay for up and downlink delay for Player2/Client2. As shown in Figs. 14 and 15, we use "ping" to validate performance of network emulator which returned averages of 19 and 39 ms.


**Fig. 14.** Validating network emulator in client1

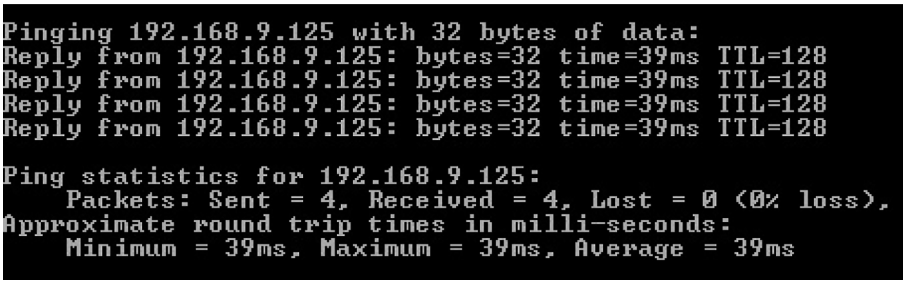


**Fig. 15.** Validating network emulator in client2

Figure 16 shows the analysed data for two players under these emulated network environments. It can be seen that the actual delays as measured by modified RTCP code are not 10/10 and 20/20 as implemented by the emulator. The delays are closer to those seen above under zero emulated delays plus the emulated delays plus additional noise caused by non-determinism in various application software and OS stack etc. This results for example in s2c delays for Player 1 of 15/19 ms v/a, c2s delays of Player 1 15/12 v/a, and for Player 2, s2c v/a delays of 26/30 ms and c2s delays of 24/26 ms v/a.

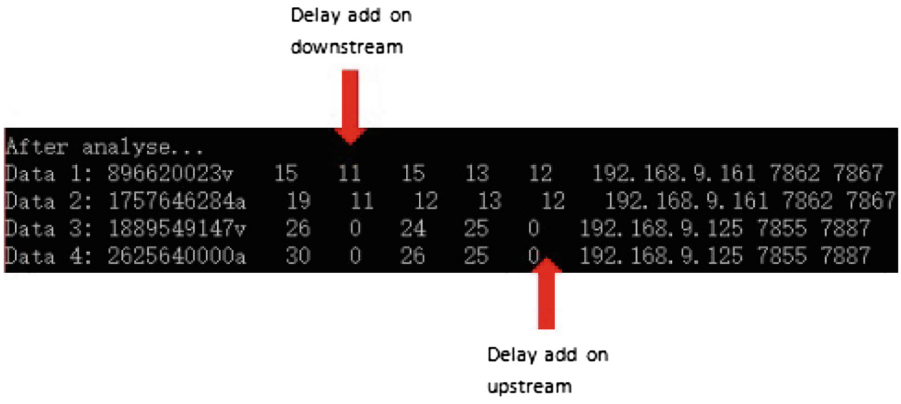


**Fig. 16.** Analysed data with different network environments

Based on these results, GA server1 requires:

- additional delay of 11 ms on downstream s2c v/a stream so that Player 1 has a total of 26 ms (15 + 11) for video and 30 (19 + 11) for audio – same as Player 2,
- additional delay of 12 ms on upstream so that totals are 25 ms – same as Player 2.

We then changed network latency to 30 ms (15 ms up/downstream) and 60 ms (30 ms up/down stream) for Player 1 and 2 respectively. The results are shown in Fig. 17.

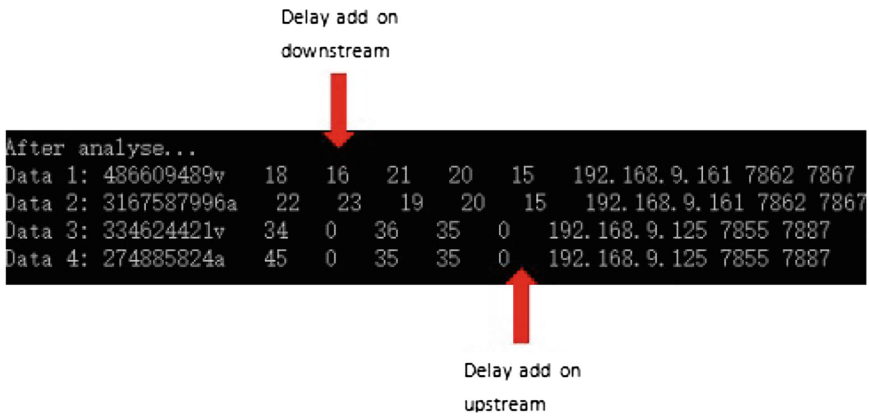


**Fig. 17.** Analyzed data after changed network latency

As above, the log server detects the changed network conditions and communicates the appropriate changes to respective streams back to GA servers in order to equalise delays.

Figure 18 shows the full set of emulated delays implemented for 6 different tests and Fig. 19 illustrates the resulting analysed data.

| Test Index: | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| P1 Up | 10ms | 20ms | 30ms | 40ms | 30ms | 20ms |
| P1 Down | 10ms | 20ms | 30ms | 40ms | 30ms | 20ms |
| P2 Up | 20ms | 30ms | 40ms | 30ms | 20ms | 10ms |
| P2 Down | 20ms | 30ms | 40ms | 30ms | 20ms | 10ms |

**Fig. 18.** Emulated delays: Test Index 0–5

```
0 Player 1: 1258346538v  11  15  26  19  19  8  27  192.168.9.161
  Player 1: 4256893225a  15  11  26  20  19  8  27  192.168.9.161
  Player 2: 1980165216v  26  0  26  32  27  0  27  192.168.9.125
  Player 2: 1988824901a  26  0  26  23  27  0  27  192.168.9.125
1 Player 1: 3953753889v  21  16  37  28  28  4  32  192.168.9.161
  Player 1: 251471564a  23  15  38  28  28  4  32  192.168.9.161
  Player 2: 1976878339v  37  0  37  32  32  0  32  192.168.9.125
  Player 2: 1372099589a  38  0  38  32  32  0  32  192.168.9.125
2 Player 1: 1990930858v  31  18  49  43  41  0  41  192.168.9.161
  Player 1: 3593159552a  30  19  49  40  41  0  41  192.168.9.161
  Player 2: 3604094500v  49  0  49  40  40  1  41  192.168.9.125
  Player 2: 2633676880a  49  0  49  40  40  1  41  192.168.9.125
3 Player 1: 1546419912v  37  2  39  51  51  0  51  192.168.9.161
  Player 1: 2130822271a  40  0  40  51  51  0  51  192.168.9.161
  Player 2: 1360988415v  39  0  39  31  30  21  51  192.168.9.125
  Player 2: 755143909a  39  1  40  30  30  21  51  192.168.9.125
4 Player 1: 1374147380v  30  0  30  39  39  0  39  192.168.9.161
  Player 1: 2376366216a  30  0  30  40  39  0  39  192.168.9.161
  Player 2: 40070154v  27  3  30  23  23  16  39  192.168.9.125
  Player 2: 479324142a  27  3  30  23  23  16  39  192.168.9.125
5 Player 1: 177863210v  21  0  21  30  30  0  30  192.168.9.161
  Player 1: 825106360a  23  0  23  30  30  0  30  192.168.9.161
  Player 2: 2156336031v  14  7  21  15  15  15  30  192.168.9.125
  Player 2: 871630376a  14  9  23  16  15  15  30  192.168.9.125
```

**Fig. 19.** Analysed data for tests (Index 0–5) with different network latency

The data (left to right) has two additional columns outlining total s2c and c2s delays (measured + compensation delays). The columns are:

- SSRC,
- s2c delay,
- delay added to downstream s2c,
- *total s2c delay after lag equalization*,
- c2s delay,
- avg c2s delay,
- delay added to downstream c2s,
- *total c2s delay after lag equalization* and
- IP address.

In each case, the mechanism is seen to work correctly by firstly detecting the impact of the emulated delays (including noise) typically within one second and then implementing lag compensation so that total c2s and s2c delays for player 1 and 2 are equal.

### 5.3 Preliminary Subjective Testing

Although the initial plan was to setup all GA servers in virtual machines for subjective testing, some technical problems and limitations arose, and thus the tests were carried out using separate servers. Since Assault Cube can supports a maximum of 5 players in a LAN, we thus used 10 computers (5 as GA servers and 5 as GA clients) to run the game. As shown in Fig. 20, a multiplayer game was created in Server1 and others joined the game created by Server1. The GA connection between GA servers and GA clients was then established. NTP was also configured in both GA servers and clients to synchronise system time. Finally, the network emulator was used to introduce artificial delay to implement different network environment.

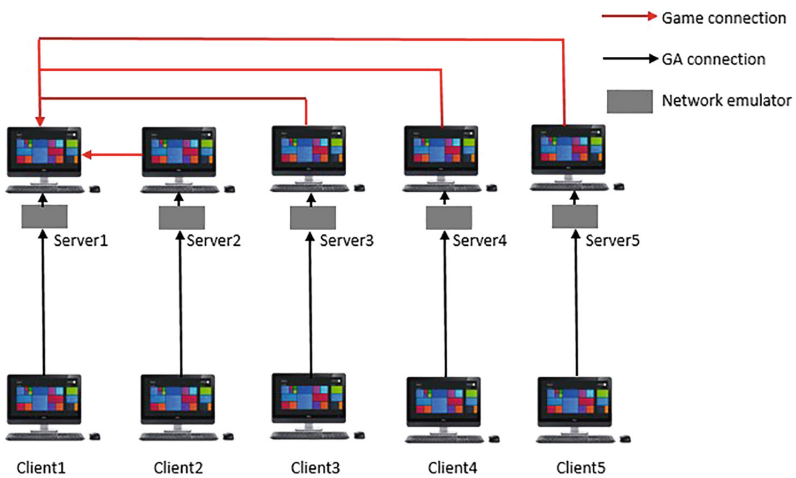


**Fig. 20.** Testbed

**Test Scenarios**

10 people were divided into two groups (5/5) to perform the test. All of the participants were postgraduate students and were familiar to varying degrees with gaming. As shown in Fig. 21, each group (Player 1 - Player 5) firstly played the game without any emulated network delay, thus providing a baseline for the tests. A series of delay scenarios (10 in total - uplink/downlink in ms) were then introduced for each player both with and without lag compensation with each scenario lasting 3 min. After each scenario, players were given a small amount of time to fill out a questionnaire and report the overall QoE and game fairness on a scale of 1–5. Both groups underwent the same series of scenarios.

| P1 | P2 | P3 | P4 | P5 | Lag Compensation |
|---|---|---|---|---|---|
| 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | No |
| 40/40 | 0/0 | 10/10 | 20/20 | 30/30 | No |
| 40/40 | 0/0 | 10/10 | 20/20 | 30/30 | Yes |
| 0/0 | 10/10 | 20/20 | 30/30 | 40/40 | No |
| 0/0 | 10/10 | 20/20 | 30/30 | 40/40 | Yes |
| 10/10 | 20/20 | 30/30 | 40/40 | 0/0 | No |
| 10/10 | 20/20 | 30/30 | 40/40 | 0/0 | Yes |
| 20/20 | 30/30 | 40/40 | 0/0 | 10/10 | No |
| 20/20 | 30/30 | 40/40 | 0/0 | 10/10 | Yes |
| 30/30 | 40/40 | 0/0 | 10/10 | 20/20 | No |
| 30/30 | 40/40 | 0/0 | 10/10 | 20/20 | Yes |

**Fig. 21.** Test scenarios

**Test Results**

The overall QoE in absence of lag equalization reported by participants is shown in Fig. 22. Whilst the sample size is small, the bar chart clearly shows that the overall QoE decreased with increasing delay which is in line with other studies on FPS games. Figure 23 illustrates the perceived game fairness for different emulated delay scenarios with and without lag compensation. Again, whilst preliminary and based on a small sample size, the results clearly show that in absence of lag compensation, higher

relative network latency results in lower game fairness. However, the game fairness remains high once lag compensation is introduced. It is very interesting to note that there was no decrease in QoE as emulated delays increased, presumably as all participants were experiencing the same delays and values were less than the 100 ms threshold that is reported as being the threshold for acceptability in FPS games.

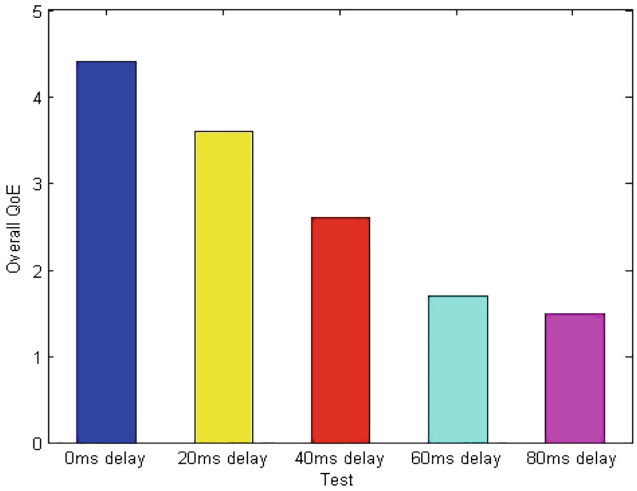


**Fig. 22.** Average overall QoE reported for different test scenarios. Delays portrayed on the x axis indicate RTTs.

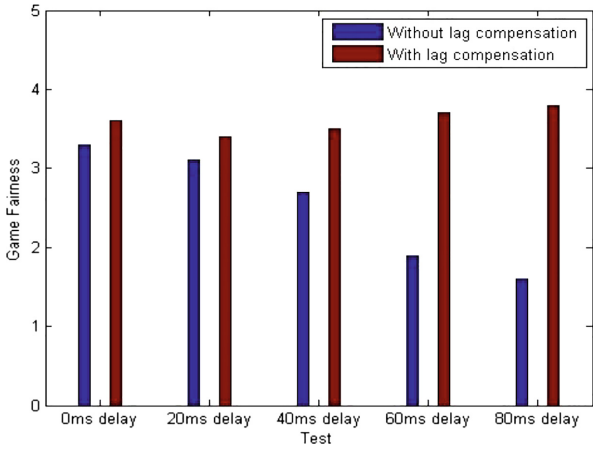


**Fig. 23.** Perceived game fairness

More comprehensive and rigorous tests are planned to fully evaluate the effectiveness of our approach. More detailed results from the above preliminary tests are available in [22].

### 5.4 Next Step – Virtualization

The above tests were carried out using dedicated servers for each player GA instance. In order to scale up the testbed, we plan to run all GA server instances using Virtual Machines (VM). This will also help to eliminate some of the non-determinism seen above in the results. This architecture is shown in Fig. 24, whereby each GA server runs in a dedicated VM and sets up connections with each client.

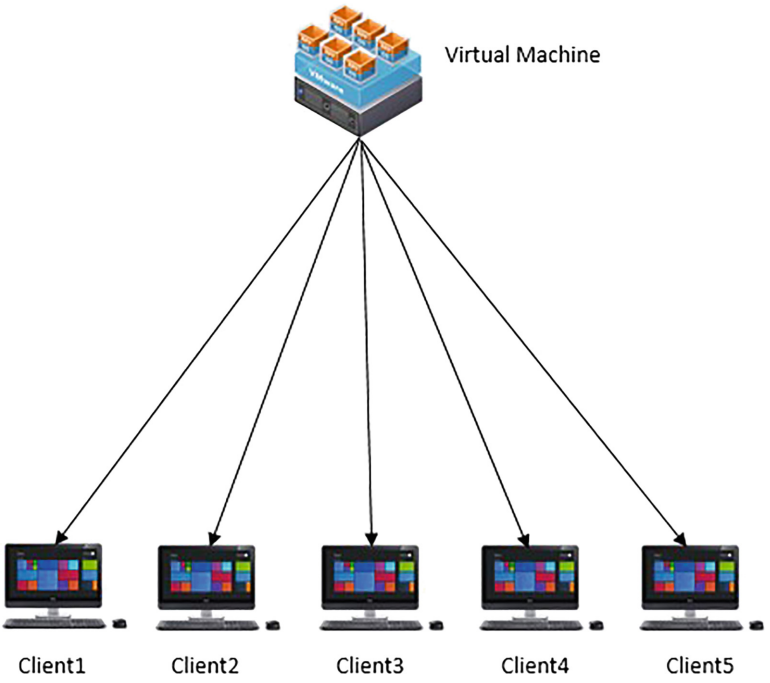


**Fig. 24.** Run GA server instances in a VM

## 6 Conclusions

In this chapter, we examine cloud gaming from the delay (lag) perspective, and in particular the impact of lag on QoE of gamers. Cloud gaming is an emerging service, which combines cloud computing and online gaming. It opens a promising direction for the computer games industry but several challenges still remain to provide good QoE for every player. We review the literature and then analyze certain QoS-related key factors and characteristics, which influence the QoE for cloud gaming, especially for so-called FPS games. The conclusion is that for FPS games, network latency presents one of the most important QoE factors. In this context, we propose a lag equalization strategy to level the playing field in the context of QoE and outline two research objectives. The first objective is to examine the feasibility of implementing a real-time lag compensation mechanism for cloud gaming. To meet this objective,

we implemented a cloud gaming system with both up and downlink lag compensation based on the Gaming Anywhere platform, an open platform for researchers. The FPS game Assault Cube was used to showcase the implementation. The mechanism uses a modified version of the RTCP protocol along with NTP to ensure adequate time synchronization to yield accurate each way delays. These are communicated to a centralized monitoring service that then determines and communicates back to each server the necessary up and downlink compensation delays. The lag compensation approach was evaluated in an emulated environment whereby a series of tests were carried out with differing uplink and downlink emulated delays to emulate differing network conditions. The results validate the mechanism by successfully implementing real-time delay equalization. To meet objective 2, we then carried out preliminary subjective tests with a small group of participants. Results firstly confirmed the impact of lag on QoE as detailed in the literature review and then validated our lag compensation approach whereby the reported QoE remained high for high delay values once equalization was implemented. Our future research will firstly optimize the experimental cloud gaming environment by introducing virtual machines for scalability. More comprehensive subjective QoE tests using the proposed lag compensation approach will then be undertaken to more rigorously evaluate its effectiveness.

**Acknowledgments.** This work has been partially supported by the ICT COST Action IC1304 - Autonomous Control for a Reliable Internet of Services (ACROSS), November 14, 2013–November 13, 2017, funded by European Union.

Andrej Zgank's work was partially funded by the Slovenian Research Agency (research core funding No. P2-0069).

We would also like to acknowledge the technical support received by the core GA developer Chun-Ying Huang.

## References

1. Jackson, P.: Video Game Revenue Forecast: 2017–22 (2017). https://www.ovum.com/research/video-game-revenue-forecast-2017-22/. Accessed 26 Apr 2017
2. Huang, C.Y., Chen, D.Y., Hsu, C.H., Chen, K.T.: GamingAnywhere: an open-source cloud gaming testbed. In: Proceedings of the 2013 ACM Multimedia conference (Open Source Software Competition Track), pp. 827–830. ACM, Barcelona (2013). http://dx.doi.org/10.1145/2502081.2502222
3. Claypool, M., Finkel, D.: The effects of latency on player performance in cloud-based games. In Proceedings of the 13th Annual Workshop on Network and Systems Support for Games (NetGames), pp. 1–6. IEEE, Nagoya (2014). https://doi.org/10.1109/NetGames.2014.7008964
4. Cai, W., Shea, R., Huang, C.Y., Chen, K.T., et al.: A survey on cloud gaming: future of computer games. IEEE Access **4**, 7605–7620 (2016). https://doi.org/10.1109/ACCESS.2016.2590500
5. Wen, Z.-Y., Hsiao, H.-F.: QoE-driven performance analysis of cloud gaming services. In: Proceedings of the 16th International Workshop on Multimedia Signal Processing (MMSP), pp. 1–6. IEEE, Jakarta (2014). https://doi.org/10.1109/MMSP.2014.6958835

6. Amiri, M., Al Osman, H., Shirmohammadi, S.: Datacenter traffic shaping for delay reduction in cloud gaming. In: Proceedings of the International Symposium on Multimedia (ISM), pp. 569–574. IEEE, San Jose (2016). https://doi.org/10.1109/ISM.2016.0124
7. Brun, J., Safaei, F., Boustead, P.: Fairness and playability in online multiplayer games. In: Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC 2006), pp. 1199–1203. IEEE, Las Vegas (2006). https://doi.org/10.1109/CCNC.2006.1593228
8. Zander, S., Armitage, G.: Empirically measuring the QoS sensitivity of interactive online game players. In: Proceedings of the Australian Telecommunications Networks and Applications Conference (ATNAC 2004), pp. 511–518. ATNAC, Sydney (2004)
9. Zander, S., Leeder, I., Armitage, G.: Achieving fairness in multiplayer network games through automated latency balancing. In; Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, pp. 117–124. ACM, Valencia (2005). https://doi.org/10.1145/1178477.1178493
10. Jarschel, M., Schlosser, D., Scheuring, S., Hoßfeld, T.: Gaming in the clouds: QoE and the users' perspective. Math. Comput. Model. **57**(11–12), 2883–2894 (2013). https://doi.org/10.1016/j.mcm.2011.12.014
11. Möller, S., Pommer, D., Beyer, J., Rake-Revelant, J.: Factors influencing gaming QoE: lessons learned from the evaluation of cloud gaming services. In: Proceedings of the 4th International Workshop on Perceptual Quality of Systems (PQS), TU-Berlin, Vienna, Austria, pp. 1–5 (2013)
12. Slivar, I., Skorin-Kapov, L., Suznjevic, M.: Cloud gaming QoE models for deriving video encoding adaptation strategies. In: Proceedings of the 7th International Conference on Multimedia Systems (MMSys 2016), pp. 18:1–18:12. ACM, Klagenfurt (2016). https://doi.org/10.1145/2910017.2910602
13. Bernier, Y.W.: Latency compensating methods in client/server in-game protocol design and optimization. In: Proceedings of the Game Developers Conference, vol. 98033, no. 425 (2001)
14. Slivar, I., Suznjevic, M., Skorin-Kapov, L., Matijasevic, M.: Empirical QoE study of in-home streaming of online games. In: Proceedings of the 14th Annual Workshop on Network and Systems Support for Games (NetGames), pp. 1–6. IEEE, Nagoya (2014)
15. Clincy, V., Wilgor, B.: Subjective evaluation of latency and packet loss in a cloud-based game. In: Proceedings of the 10th International Conference on Information Technology: New Generations (ITNG), pp. 473–476. IEEE, Las Vegas (2013). https://doi.org/10.1109/ITNG.2013.79
16. Jarschel, M., Schlosser, D., Scheuring, S., Hoßfeld, T.: An evaluation of QoE in cloud gaming based on subjective tests. In: Proceedings of the 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2011), pp. 330–335. IEEE Computer Society, Washington, DC (2011). http://dx.doi.org/10.1109/IMIS.2011.92
17. Amiri, M., Malik, K.P.S., Al Osman, H., Shirmohammadi, S.: Game-aware resource manager for home gateways. In: Proceedings of the International Symposium on Multimedia (ISM), pp. 403–404. IEEE, San Jose (2016). https://doi.org/10.1109/ISM.2016.0091
18. Claypool, M., Claypool, K.: Latency and player actions in online games. Mag. Commun. ACM **49**(11), 40–45 (2006). https://doi.org/10.1145/1167838.1167860
19. Beyer, J., Varbelow, R., Antons, J.N., Zander, S.: A method for feedback delay measurement using a low-cost arduino microcontroller: lesson learned: delay influenced by video bitrate and game-level. In: Proceedings of the 7th International Workshop on Quality of Multimedia Experience (QoMEX), pp. 1–2. IEEE, Pylos-Nestoras (2013). https://doi.org/10.1109/QoMEX.2015.7148095

20. Amiri, M., Osman, H.A., Shirmohammadi, S., Abdallah, M.: Toward delay-efficient game-aware data centers for cloud gaming. ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) **12**(5), 71/1–71/19 (2016). https://doi.org/10.1145/2983639
21. Huang, C.-Y., et al.: GamingAnywhere: the first open source cloud gaming system. ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) **10**(1), 10/1–10/25 (2014). https://doi.org/10.1145/2537855
22. Li, Z.: Time Aware Gaming – Levelling the playing field for everyone. In: MSc thesis, National University of Ireland, Galway, Ireland, August 2017. Available on Request