The effects of relative delay in networked games

Tristan Nicholas Hoang Henderson

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University of London.

Department of Computer Science

University College London

February 2003

Abstract

Games are one of the most popular multiuser applications currently in use on the Internet. They have become so in spite of the lack of **Quality of Service** (QoS) guarantees offered by the current Internet, which are typically believed to be a requirement for delay-sensitive multimedia applications such as games. Understanding how networked games have become popular is therefore important for designing applications that can become successful with or without the presence of QoS guarantees.

One reason for the popularity of games may be the interaction between players in a multiuser game. It is this interaction that compels users to play a networked game, since without other players there is little benefit to the networked component of the game. Players may be willing to tolerate lower QoS if they are able to enjoy a game with other users.

This thesis examines users' preferences for one QoS parameter, delay, in networked First Person Shooter (FPS) games. We consider a player's **absolute delay** (the delay between a player and the game server), and their **relative delay** (the difference between a user's delay and that of the other players). We employ controlled and uncontrolled objective and subjective experiments: monitoring of publicly-available game servers, group experiments, a survey of game players, and controlling the delay to servers for the FPS game, *Half-Life*.

We find that users are drawn to game servers where they can interact with a greater number of players. Delay has a greater effect on a player's decision to join a game than to leave, and a player's tolerance for delay increases with the time that they remain in the game. Although they believe relative delay to be important, in practice users are more concerned about absolute than relative delay, and can find it difficult to accurately distinguish their relative delay.

Acknowledgements

Although written in the first person nominative plural personal form (also known as “we”), this thesis is all my own work. It would never have been finished, however, without the help of several people, who I acknowledge here.

Saleem Bhatti and Jon Crowcroft have gone beyond the call of duty in providing guidance and encouragement.

I am grateful to Grenville Armitage for giving me the idea of running my own game servers. Hopefully we will work together one day! John Andrews was invaluable in helping me set up said game servers at UCL.

Thanks to Huw Oliver, Erik Geelhoed and Mike Spratt at HP Labs, without whom the experiments in Chapter 6 would never have taken place, and to Brendan Murphy at Microsoft Research for his help with the experiments in Chapter 4. Colin Perkins provided access to a host at ISI East. I must also thank the European Leisure Software Publishers Association (ELSPA) and the Internet Domain Survey (IDS) for allowing me to use their data.

Thanks to Jörg Widmer, Martin Mauve and Lars Wolf for convincing me that games are a worthwhile area of research!

Thank you to Angela Sasse and Søren Sørensen for getting me through the UCL PhD process.

Thanks to Ian Brown, Piers O’Hanlon, Ken Carlberg, Manuel Oliveira, Panos Gevros and of course the staff of the JB. I might have got the thesis in earlier if not for them, but it wouldn’t have been half as much fun. . .

Last, but certainly not least, thank you to Vicky.

Note on references

The nature of the networked gaming industry and community means that several of the sources referred to in this dissertation exist only on the World Wide Web. All Universal Resource Identifiers (URIs) have been checked, but their longevity cannot be guaranteed. Where appropriate, the bibliographic references contain a date of citation, as recommended by ISO standard 690-2 [99], that indicates when the URI of the article in question was checked and found to be available.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The latter half of the 1990s has seen explosive growth in the Internet, in terms of the number of connected nodes and users. The Internet has evolved from a primarily research-oriented network to a part of everyday life for millions of people around the world. There has yet to be a commensurate growth in the number of applications and services, however, and the predominant uses for the Internet have remained applications which were designed over a decade ago, such as e-mail and the World Wide Web [78, 150, 149].

It has been argued that this dearth of new applications is due to the **best-effort nature** of the Internet. Applications such as streaming audio and video, or multimedia conferencing, have stringent requirements that cannot be met by a network that cannot guarantee throughput and delay. The Internet is such a network, due to the finite amount of bandwidth that is available, and the nature of the Internet Protocol (IP), which is connectionless and offers no admission control or flow state. For those flows with specific requirements, it is therefore necessary to provide the ability to differentiate between application flows such that these requirements can be met. Without this, multimedia real-time applications will be unusable. It is the provisioning of these applications that has driven efforts such as IntServ [33] and DiffServ [26] to develop methods to enable **Quality of Service** (QoS) in the Internet.

One real-time multimedia application that has become popular, in spite of the lack of network QoS guarantees offered by today's Internet, is the multiplayer networked game. Games contribute to an increasingly large proportion of network traffic [138], and the online games industry is predicted to be worth US$5 billion by 2004 [54]. Networked gaming is considered a *bona fide* sport by some, and international gaming tournaments have made it possible to be a professional game player [48]. Network usage by game players is likely to increase further, as the types of end-systems capable of playing networked games continue to proliferate. In addition to the existing personal computers (PCs) that are capable of playing networked games, games consoles such as the Sega Dreamcast [179], Microsoft Xbox [142] and Sony PlayStation

2 [188] have begun to feature Ethernet and telephone connectors, and the games designed for these appliances also include networked capabilities. Digital set-top boxes for television such as those employed by Sky Digital [185] also feature multiuser gaming applications. These games consoles and set-top boxes are mass-market, single-purpose devices, and are generally priced well below the cost of a PC. As such, many more households have access to a console or set-top box as opposed to a PC, and so the introduction of these new networked devices can be expected to increase the number of online game players.

The success of games is even more intriguing, because they are one of the most obvious applications to require a higher level of network QoS. Certainly, game players are already willing to pay extra to get an improvement in their playing experience, as evidenced by specialist gaming hardware such as joysticks, mice, mousepads and even furniture. Many game players are willing to spend much more on computing hardware than the average computer user, and specialist manufacturers have emerged to take advantage of this [14]. Game publishers have also proposed new methods of capitalising on players' willingness to pay, for instance by charging players each time they play a game via a network, rather than the current practice of charging a one-off fee for the software [159, 140]; by charging a fee per game with the opportunity for players to win money or prizes [196]; by charging to allow players to advance through a game more quickly [195]; or even by paid product placement within a game's virtual world [23]. More interesting, from a networking point of view, is the existence of modems marketed as being specially optimised for games [1], and software designed to determine network characteristics of potential game servers such as delay [75]. These developments indicate that game players are interested in network QoS, and would be willing to pay for the ability to improve it. Yet, QoS is evidently not a huge barrier to game deployment, since millions of copies of networked games are already being sold and played.

The popularity of games as a group application merits their study, and has potential benefits in various areas. One of the perennial reasons given for the lack of multicast deployment, even after over a decade of research, is the lack of appropriate applications. Internet Service Providers (ISPs) will not enable multicast in their networks, until their customers demand it. Their customers will not desire multicast, or perhaps even be unaware of its existence, until there are compelling applications that require it. By the same token, however, application authors may be reluctant to design multicast applications until multicast is ubiquitous. One approach to resolving this "chicken and egg" problem may be to examine the currently-available multiuser applications, such as games.

It is important to understand the network QoS requirements for games, given their popularity in spite of the lack of network QoS guarantees. There is a large body of work analysing the QoS requirements for other networked applications. The first experiments with voice over a packet network took place as early as 1973, leading to the Network Voice Protocol [46]. The requirements for voice traffic are now well-understood, and the subject of international standards such as ITU-T G.114 [102]. Networked video has also been the study of much research, and there are standards for video conferencing, e.g. H.323 [100]. Games, on the other hand, have received a disproportionately small amount of interest from the networking research community or standards bodies.

When considering network QoS, there are several parameters that are usually considered, such as throughput, loss, and jitter. Of these, **network delay**, or "lag" as it is sometimes referred to in game-playing circles, is the most commonly-cited concern of game players. There are websites dedicated to users' complaints about delay and possible remedies for high latency [115]. In online discussion forums such as Slashdot [186], there are many statements from users such as:

- "150 [ms] is not tolerable."
- "no way anybody can play quake competitivly [*sic*] with a ping over 200ms."
- "In Q3 [*Quake III*] I can't play with a ping over 90 [ms]"
- "I find a ping of more than 50 [ms] intolerable. I won't play a game at 100 [ms] or more."

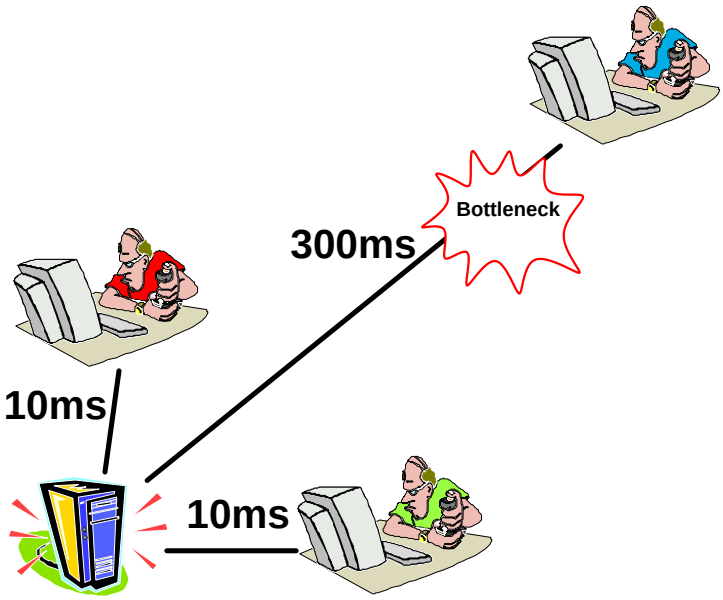
It would thus appear that users believe the delay bound for real-time multiplayer games to be very low indeed. Yet the *prima facie* evidence is that users are playing games even when these low delay requirements cannot be met. Why is this?

The Internet is usually portrayed as a "tragedy of the commons" [83], with greedy and selfish users all competing for the same shared scarce network resource, resulting in overconsumption and little network capacity being available for anyone [130]. This might be true if the individual users are all unrelated, for instance where they are all using different, single-user, applications. Does the assumption of selfishness necessarily hold, however, for group applications? If the members of a group shared a common network bottleneck, might these members consider the effects of their actions on each other? Might they be less selfish towards users with whom they are interacting? Games are inherently a social activity — Shelley defines a game as "a series of interesting decisions in a competitive environment" [180]. Whilst a computer-generated set of opponents can potentially provide a competitive environment, arti-

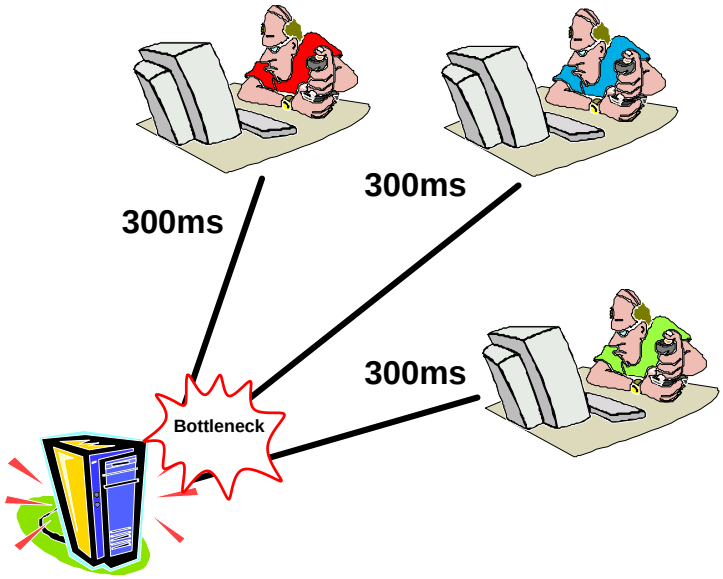
ficial intelligence has not evolved to the stage where a widely-available personal computer can appropriately simulate human judgement. Networked computer games thus generally entail interaction with other users. Perhaps it is this interaction that can explain the popularity of games where other multimedia applications have failed.

If we accept that users in a game will consider the other players of the game, then perhaps they will consider each others' network conditions. Delay has been mentioned as a primary concern of players — the client-server network topology of most networked games means that the network delay is that between the client, or player, and the host which is running the game server daemon. Consider the following two scenarios:

- **Scenario 1a:** Three users are connected to a game server. Two users are on the same local area network as the game server, and as a result they each have a low round-trip time (RTT) of 10ms between them and the server. The third user, however, is connected via a high-latency wide-area network link, and there is a bottleneck between this client and the game server, resulting in an RTT of 300ms.



- **Scenario 2a:** Three users are connected to a game server. There is a bottleneck near the server — this could comprise network congestion near the server, or perhaps the server's CPU is heavily loaded, resulting in a delay in processing time. For the purposes of this discussion, the causes of delay are not so important as the effects — that is, the type of gameplay that the user experiences as a result of the delay. In this scenario, all the users have an RTT of 300ms, and consequently experience degraded gameplay.



In Scenario 1a, two of the users have low delay, but have chosen to play with another user with a much higher delay than themselves. We can assume that they want to play with this high-delay user, and vice versa. But the nature of the game will change due to their differing delay characteristics. The low-delay players may be able to respond quicker than the high-delay user, since they may receive information from the server sooner, which could be necessary to perform well in the game. This imbalance may diminish the enjoyment that all the players receive from the game, as the competitive environment which defines the game is no longer truly competitive, but biased towards some players.

In Scenario 2a, although the average delay experienced by the players is higher, the competitive environment, at least with respect to delay, is maintained — there is a level playing field. Might users prefer this level playing field, albeit with a higher response time? There is reason to believe that this might be the case; returning to the previously-mentioned online discussion forum [186], we find comments from users such as:

- "I liked it better when it was a more even ping spread."

- "I don't find ping ever to be a problem. The issue is the relative pings. If everyone was put to the same [dis]advantage pings wouldn't matter."

This thesis aims to contribute to our understanding of multiuser networked computer games by carrying out an analysis of a particular genre of game, the first person shooter (FPS), in which network conditions such as delay could be tested. The following questions are examined:

- **Q1** Do players in a multiplayer game consider the presence of other players when choosing where and when to play?

- **Q2** Do players in a multiplayer game consider the network conditions of other players when choosing where and when to play?

1.1 Thesis

We have proposed that users may potentially consider other game players and their network conditions when choosing to play a multiplayer networked game. As delay is perhaps the most important QoS parameter in a game, we concentrate on this. Other parameters such as bandwidth are out of scope of this work (the reasoning for this is detailed in Chapter 2). We offer the following thesis:

> **Users prefer similar relative delay, rather than minimal individual delay, in networked games, and will self-organise with respect to the other users in a game to achieve this.**

1.2 Goals and approach

This dissertation attempts to demonstrate that users consider the network latency of other users on a game server when playing a multiplayer networked game. We wish to answer the questions listed above, Q1 and Q2. In order to do so, our primary approach is to analyse "real" users, that is, players of existing games, using existing game server software. While the development of a custom game for research purposes has many benefits, such a game will never be as polished or finished as a commercial game, since commercially-produced games typically have a much larger team of developers than a Ph.D project. The user experience might be different for a game developed for research, since part of the player's experience comes from the detailed graphics, sounds and so on, which are present in a commercial game. Moreover, without this additional level of completeness, the potential number of users of a game developed for research will be much lower than that of a commercial game, as the game may appear less attractive.

To examine Q1, we monitor users on a variety of game servers. There are thousands of game servers accessible via the Internet at any given time. By analysing the session-level characteristics of users on these servers, we develop a model for session-level user behaviour. Using this model, we can examine the relationship between the number of users on a server, and the decision of other users to join or leave that server. This can be seen as an uncontrolled objective methodology.

To examine Q2, we run our own game servers. Monitoring publicly-available servers necessitates a lower resolution of data, due to the difficulties of retrieving and collecting data from a disparate set of distantly-connected hosts. By using our own servers, we can conduct detailed analysis of the players on the server. It also allows the manipulation of network conditions by making adjustments at the network packet level to communications with client systems. This can be seen as an controlled objective methodology.

We also carry out subjective measurements. A questionnaire survey of game players is carried out — an uncontrolled subjective study. Finally, a controlled subjective methodology is used: human factors experiments involving users' perceptions of absolute and relative levels of network delay in a game scenario.

1.3 Outline of dissertation

In Chapters 2 and 3 of this dissertation we examine the motivation and the context of this work, and examine related work and the state of the art:

- In **Chapter 2**, we examine the evolution of networked games, as well as similar applications such as military simulations. The most common types of networked game are described. We outline the ways in which delay can affect a game-playing experience, and the methods used by games to compensate for network delay. Some relevant economic theory is considered to explain why users might consider other users when interacting in a group application such as a game.

- In **Chapter 3**, we consider related work, including network analyses of games, resource management schemes for games, and studies of user preferences for delay in multimedia applications. We outline the game which is the focus of this dissertation, and the reasoning for choosing this particular game.

In Chapters 4, 5 and 6, we document the analysis and work that has been carried out.

- In **Chapter 4**, we demonstrate that networked games exhibit **network externalities** — that users in a game consider the existence of the other users in the game when choosing to connect to a game server.

- In **Chapter 5**, we examine the network delay characteristics of users on a publicly-available game server. By passively monitoring the delay of players connecting to the server, and by inserting additional delay into players' flows, user preferences about delay are inferred and explained in the context of the game itself.

- In **Chapter 6**, we examine game players' preferences and reactions towards network delay. Using a questionnaire and laboratory experiments, we consider the performance of FPS game players under different levels of network delay, to see what levels of delay players notice, respond to, and prefer.

Finally, in **Chapter 7**, we conclude the dissertation with a summary of the main contributions of this research, and discusses the potential research that could be carried out in the future.

Chapter 2

The evolution of networked games

In this chapter, we consider the way that networked games are evolving and what this will mean for the use and provisioning of networked games in the future. We provide a short history of networked gaming, and the other applications, such as military simulations, which have contributed to the development of networked multiplayer games. We move on to discuss some of the QoS requirements for games, and the networking techniques that games developers use to deal with problems in the network. Finally, we discuss how the interaction between multiple users may affect the level of QoS required by an application, using examples and theory from economics and sociology.

2.1 Multiuser networked games and applications

This section describes the evolution of multiuser networked games, from the earliest computer games of the 1950s and 1960s to the large-scale virtual worlds of today. The most popular different types of networked games are described. Networked games share features with other networked applications, and these related applications are discussed.

2.1.1 Networked Virtual Environments

Networked games are a subset of the genre of applications known as **Networked Virtual Environments**. An NVE is defined as "a software system in which multiple users interact with each other in real-time, even though those users may be located around the world" [184]. Before discussing networked games, it is useful to describe the characteristics, in particular the communication models, of NVEs.

Singhal [184] states that an NVE is distinguished by five features:

1. A shared sense of space
2. A shared sense of presence
3. A shared sense of time

4. A way to communicate

5. A way to share

The first three features can be viewed as the senses that an NVE creates, whereas the latter two features are the mechanisms that enable these senses.

An NVE generally comprises three components: a database representing the shared virtual world, a communications substrate, and a set of end devices on which to display the contents of the world. Users, or **players** [27], can manipulate the **entities** in the world, and in doing so generate **events**. For most NVEs, the virtual world is three-dimensional, and players are responsible for controlling characters or avatars and interacting with other entities in the world. The purpose of the communications substrate is to keep all of the users aware of the current state of the world.

NVEs use one, or a combination [73], of two network topologies: **client-server**, e.g. MASSIVE [79], or **peer-to-peer**, e.g. DIVE [70].

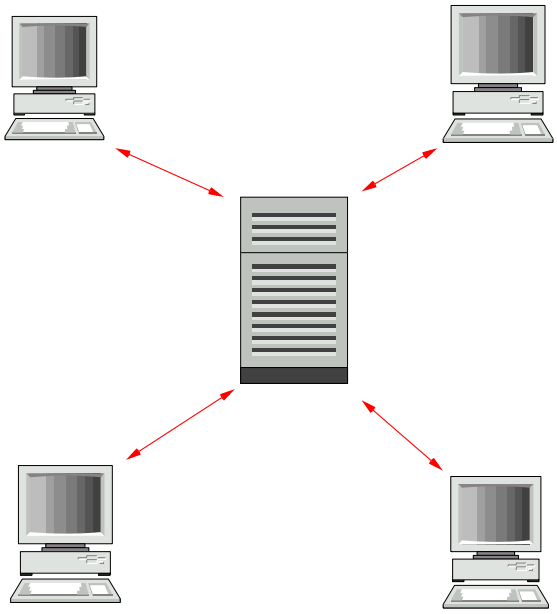


Figure 2.1: Client-server network architecture

In a client-server architecture (Figure 2.1), one host acts as a server. This server is responsible for maintaining the database of client state and thus the state of the virtual world. Users who wish to participate in the NVE connect, usually via unicast, to this server. All network messages between the players are transmitted to the server, and the server then propagates the messages to all the other players. A client may also exist on the same host as the server, in which case the server is referred to as a **non-dedicated server**.

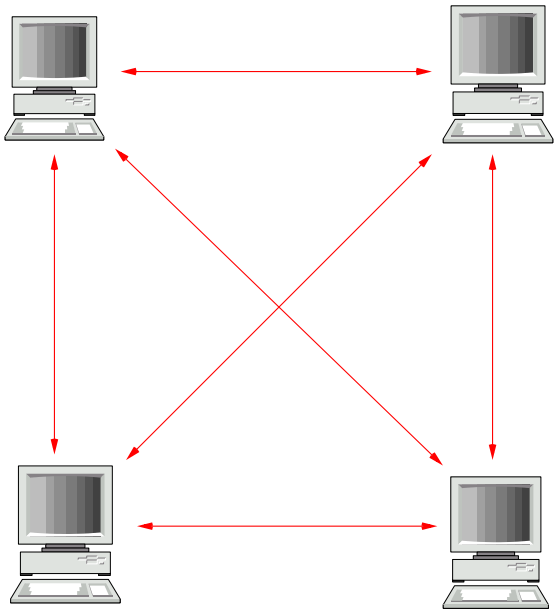


Figure 2.2: Peer-to-peer network architecture

In a peer-to-peer architecture (Figure 2.2), there is no dedicated server. The database is fully distributed amongst all the participants, and all the players are responsible for keeping track of the state of the world. Messages are sent from each player to all the other players.

2.1.2 Military Simulations

One of the most common NVEs is the military simulation. SIMNET [5] was designed for training tank operators. It used a distributed peer-to-peer architecture, and all of the objects would broadcast events to all the other participants. Distributed Interactive Simulation (DIS) [164] is a formalisation of the SIMNET protocols, and has been standardised by the IEEE [96]. DIS was designed to be able to simulate many military tasks, such as training operators of tanks, aircraft and ships, and multiple types of vehicles can participate in the same simulation. The state of entities and events is transmitted between participants using Protocol Data Units (PDUs).

Current military simulation research revolves around the High Level Architecture (HLA), which is also an IEEE standard [97]. HLA does not specify any particular technology, but instead provides a generalised architecture for distributed simulation. HLA is designed so that it could be used in non-military simulations as well.

2.1.3 Computer-Supported Cooperative Work

Computer-Supported Cooperative Work (CSCW) is an area of research defined as “an endeavor to understand the nature and requirements of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements” [178]. Schmidt and Bannon go on to define cooperative work as that where “people engage in *cooperative work* when they

are *mutually dependent* in their work and therefore are required to cooperate in order to get the work done". In a loose sense, then, a game could be viewed as a form of cooperative work, since if the other players do not play, the game cannot be played, and need not exist. On the other hand, a game is a competitive environment, where players are generally aiming to defeat their fellow participants. The nature of the task is not collaborative, and this might create differences in behaviour between users of these two types of applications. This has been shown to occur in computer games when players are asked to cooperate, or to compete [7].

Despite this difference in tasks, CSCW and games are closely related. CSCW can encompass such tools as audio- and video-conferencing, shared whiteboards, e-mail, NVEs, and many other applications, all of which are commonly identified in CSCW terminology as **groupware**.

2.1.4 Networked games

The most popular form of NVE is the multiplayer game. These have existed for as long as the computer game genre itself. Table 2.1 lists some of the important multiplayer networked games that are mentioned in this section.

In 1958 William Higinbotham created what is generally accepted [8, 3] as the world's first computer game. *Tennis for Two* ran on a dedicated analog computer that Higinbotham built from spare parts in his place of employment, the Brookhaven National Laboratory in New York. As its name implies, the game was for two players — a ball was displayed on an oscilloscope's screen, and players turned a knob and pushed a button to move and hit the ball.

In 1961, Steve Russell wrote the game *Spacewar* for the PDP-1 computer located at MIT [125]. This was a space simulation where players controlled spaceships, and the object of the game was to shoot at and destroy the other ships. *Spacewar* was also a multiplayer game, designed for two players. In 1969, Rick Blomme ported *Spacewar* to the Xerox PLATO time-sharing system. The two players could now play on remote terminals, and *Spacewar* thus became the first multiplayer networked game.

In 1978 Roy Trubshaw and Richard Bartle created the first MUD (Multi-User Dungeon) [18]. This was one of the first games to feature a shared virtual environment in the form of a database, in which multiple players could simultaneously interact with each other, as well as with the world itself. Players remotely logged into a DECsystem-10 computer that was located at the University of Essex. The objective of the game was to accumulate points, by picking up objects and treasure, or by killing the other players. Users could also add to the database, thus augmenting the environment, via a programming language.

From the original MUD, networked multiplayer games have since developed into three main categories: the **First Person Shooter (FPS)**, the **Real-Time Strategy Game (RTS)** and

Date	Game or System
1958	*Tennis For Two* — the first computer game
1961	original *Spacewar*
1969	multiplayer *Spacewar* — the first multiplayer networked game
1974	*Maze* — the first FPS game
1979	MUD 1 — Multi-User Dungeon
1985	*Amaze*
1985	*Dogfight* — multicast flight game for SGI workstations
1987	*MIDI Maze* — a multiplayer FPS using MIDI for networking
1992	*Castle Wolfenstein 3-D*
1993	*Doom* — the first popular networked FPS game
1994	*Doom II*
1995	*CivNet* — one of the first “Massive” MMORPGs
1996	*Quake* — the first popular client-server FPS game
1997	*Quake II*
1998	*Diablo II*
1998	*Half-Life* — the subject of this dissertation
1998	*MiMaze* — multicast distributed game
1998	Sega Dreamcast - games console with modem for playing MMORPGs
1999	*Unreal: Tournament*
1999	*Quake III Arena*
2000	Sony Playstation 2
2001	Microsoft Xbox — first games console with built-in Ethernet

Table 2.1: Dates of important games and gaming systems

	FPS	**RTS**	**MMORPG**	**non-real-time**
Architecture	Client-server	Client-server / peer-to-peer	Client-server	Peer-to-peer
Number of servers	High	Low	Low	None
Players per server	Low	Low	High	—
Persistence of world	Short-lived	Short-lived	Long-lived	—
Examples	*Quake*, *Half-Life*	*Starcraft*, *Age of Empires*	*Everquest*, *Ultima*	Chess

Table 2.2: Types of networked game

the **Massively Multiplayer Online Role-Playing Game (MMORPG)**. Williams attributes the dearth of game genres to Hotelling's theory of centrality and homogenisation in markets [93]: "as hit titles generate interest in a new game format, competitors copy the format, predictably more eager to split the profits for a sure thing than to risk the failure of a more innovative format that might only appeal to some smaller group" [201]. Thus, we find that the FPS, RTS and MMORPG genres account for the majority of networked games. In addition, there are a number of non-real-time games that are played across the Internet, such as chess — being non-real-time, they have quite different network characteristics and are determined to be out of the scope of this thesis. The main differences between these types of game are outlined in Table 2.2.

The MMORPG can be seen as the evolution of the graphical MUDs and virtual worlds that developed during the 1970s and 1980s. Most commercial MMORPGs use a client-server architecture — the virtual world is controlled by a server, and players connect to this server in order to play the game. Typically there are a very small number of servers, each of which services a large number of players. For instance, the Sony MMORPG *Everquest* features 400,000 regular users, each playing for an average of 20 hours a week, all serviced by a cluster of servers in California [62], whilst the Korean MMORPG *Lineage* claims that four million users play on its servers, each of which can service 150,000 concurrent users [118].

The typical task in an MMORPG is to build up a character or set of characters, collecting knowledge, weapons, money, property, etc. through interaction with other characters, such as talking or fighting, and through quests for specific goals. It can take many months for a player to build a desirable character, and secondary markets have developed where players sell characters

Figure 2.3: *Quake* — a typical FPS game

to each other for physical money [175]. With hundreds of thousands of players in the virtual world, the virtual economy of that world can grow quite large [37], and a priority for game designers is to ensure that this economy does not collapse [183].

The FPS game is believed to have originated in 1974, when Dave Lebling and Greg Thompson developed the game *Maze*. FPS games are so-called because the camera, or viewpoint, of the player, is through the eyes of the character that they are playing in the game; in other words, a first person viewpoint. Each player is responsible for controlling a single character. Other defining characteristics of the FPS genre include the exploration of a three-dimensional virtual world which is shared amongst all the players in the game, and a simple objective — shooting and destroying objects. A screenshot from a typical FPS game is shown in Figure 2.3. *Maze* used a client-server architecture — clients ran on Imlac PDS-1s, whilst the server ran on a PDP-10.

Amaze [22] was one of the first multiplayer shooting games to arise from the research community. Not strictly an FPS game (the viewpoint was from above the character), *Amaze* featured players negotiating a monster through a maze, shooting at the other monsters. The game was designed to run on the V distributed system [41] and was peer-to-peer — each player sent its location and state to each of the other players on the network.

In 1993 id Software released *Doom*, which was one of the first multiplayer FPS games for personal computers (PCs). Up to four players could play over a LAN. Each player connected to each of the other players in a peer-to-peer topology using IPX (Internetwork Packet eXchange) from Novell, a connectionless network protocol. An entity's position and state was broadcast by each player using packets of a fixed (495 byte) size. *Doom* was very successful, selling over a million copies [117] and soon became the scourge of network administrators everywhere [39] due to its broadcast nature and the lack of any congestion control or data-reduction mechanisms — the messages created by a four-player game could swamp an office's LAN. Moreover, the networking model used in *Doom* created problems for the players. The use of a lockstep mechanism, whereby players have to wait for all the other players to respond before the game's clock can advance, meant that the speed of the game was determined by the slowest responding machine.

Doom spawned many sequels and derivative games — *Doom II*, *Ultimate Doom*, *Final Doom* and so on, all of which had similar gameplay and networking characteristics to the original version of *Doom*. In 1996, however, id Software released its next generation of FPS game, *Quake*. *Quake* abandoned the broadcast network model of its predecessors and introduced a client-server architecture. All of the network communication used unicast UDP.

Quake has spawned two successors, *Quake II*, *Quake III: Arena*, which share similar network characteristics to the original *Quake*. Additionally, Valve Software licensed the *Quake II* game engine from id Software to develop their own FPS game *Half-Life*.

One of the most popular FPS games of recent years that is not based on id Software's code has been Epic Games' *Unreal Tournament*. This game is very similar to the *Quake*-based FPS games, with users shooting at each other in a shared virtual environment. Multiplayer gaming is available via a UDP-based network protocol which is similar to that of *Quake*. Another FPS game of note is Bungie Software's *Halo*. This was one of the main software titles used to launch the Microsoft Xbox games console.

The RTS game genre evolved from the tabletop war and strategy simulation games. RTS games typically involve controlling a large number of entities, such as an army. Unlike the relatively simple tasks of an FPS game, an RTS game requires the player to devise strategies to look after these entities and to complete tasks such as defeating other armies, or building successful economic communities. Some of the most popular RTS games include *StarCraft*, *Settlers* and *Age of Empires*. Most RTS games involve a small number of players, and unlike the single central MMORPG servers, the server for an RTS game usually runs on one of the players' hosts.

Games are fast becoming one of the most popular real-time networked applications on the Internet today. McCreary and Claffy [138] study 10 months worth of traffic at a major Internet peering point, and find that games account for at least[1] 14 of the top 25 UDP applications that traverse the exchange. The most popular networked games sell millions of copies — Microsoft's *Age of Empires* sold 6 million copies, and id Software has sold upwards of 8 million FPS games [117]. Wireless games are already a popular mobile application [104], and some analysts predict that wireless games will be one of the biggest sources of revenue for future generations of mobile phone networks [53], with one report projecting that hundreds of millions of people will play such games within the next five years [55].

The popularity of some of these networked games has led to network problems. The publishers and server operators of many of the large MMORPGs are often taken by surprise by the demand for their games, and it often takes several months for sufficient server capacity to be provided [176]. This may be due in part to the fact that games developers rarely give networking a high priority — customer service is considered more of a concern than networking by some developers [57]. In spite of these failings, players are still keen to access these virtual worlds.

2.2 User requirements for multimedia applications

We have already mentioned how games have become a popular networked application, in spite of the lack of QoS guarantees available from the current Internet. To understand why QoS guarantees are important for real-time applications such as games, it is necessary to examine the QoS requirements for these applications.

2.2.1 Quality of Service

Mathy *et al.* [133] list the five QoS parameters which are typically applied to group multimedia applications:

- **throughput** — the minimum data rate
- **transit delay** — the elapsed time between a data message being emitted from a sender and consumed by a receiver
- **delay jitter** — the maximum variation allowed in delay
- **error rate** — the ratio of incorrectly-received or lost data to sent data
- **degree of reliability** — the minimum number of members of the group that must receive each item of data

[1] five applications were not identified

Sales rank	**Title**	**Advertised throughput requirements**
1	*The Sims: On Holiday*	28.8 kbps
2	*Star Wars: Jedi Knight II*	56 kbps
3	*Medal of Honour*	33.6 kbps
4	*Dungeon Siege*	56 kbps
5	*FIFA 2002 World Cup*	56 kbps
6	*The Sims*	28.8 kbps
7	*The Sims: Hot Date*	28.8 kbps
8	*Championship Manager: Season 01/02*	LAN
9	*Half-Life: Generations*	28.8 kbps
10	*Zoo Tycoon*	N/A

Table 2.3: Top ten games sold in the UK and their throughput requirements, 15/05/2002

Different applications will have different requirements for each of these parameters — for instance, a video conference might require low jitter, but tolerate a high level of loss, whereas a shared whiteboard might require no loss, but tolerate low bandwidth.

Throughput is currently not a major factor in the user experience of a networked game player. Many game players connect to the Internet using dialup modems. As such, most of the existing commercially-available games have been designed for these users, and are thus capable of operating over very low-bandwidth links such as a modem connection. Table 2.3 shows the network requirements, as advertised by the respective games publishers, for the top ten games sold in the UK for the week ending 15/05/2002 [64]. Nine of the games offer a networked multiplayer option, and only one of these requires Ethernet connectivity. Throughput is therefore not a large factor in current networked games, although with the introduction of broadband DSL and cable modems, it might become so in the future.

Jitter, or the variation in delay over time, is a potential problem for game players. Obtaining an accurate view of the effects of jitter, however, is difficult, due to the overhead of taking large-scale detailed measurements of jitter. As we will explain in Chapter 5, we required a system for passively monitoring game players. Such a passive monitoring system cannot involve the installation of additional software at the client. Accurate jitter estimation, however, requires measurements on the order of milliseconds [122], and to do so without the installation of additional client software would involve a large amount of network probing. It would therefore be

Figure 2.4: *Half-Life* game server browser

very difficult to measure jitter in a passive measurement system, since the addition of this level of measurement traffic might affect the network being measured, especially if the users being measured are on low-bandwidth links.

Several researchers assert that delay is the most important parameter of performance for a networked multimedia application [52, 167, 169], and in particular for games [42, 10]. Complaints about delay by game players have already been discussed, and the importance of delay, perhaps as a result, has been incorporated into game design — in an FPS game, the server browser which helps a user to choose between a set of potential game servers displays the delay between the user and a server, but not the loss or jitter (in Figure 2.4, the circled column marked "Net Spd" is intended to provide an indication of network delay between the player and game server).

2.2.2 Group interaction

Typically, in a QoS-enabled network, each network user or application declares what will be required from the network for that application's traffic. To some extent, these requirements can be made on a technical basis — the type of application, the user's hardware and software constraints, and so on. The requirements can also be made on the basis of the preferences of the user — a user who values a particular video stream highly, for instance their favourite

TV show, might pay more to specify a higher throughput and hence video quality, then for a video stream in which they only have a passing interest. Having received the requirements, the network can then determine whether or not these can be met, and perhaps deny the user admission to the network on this basis. A network provider may choose to offer a Service Level Agreement (SLA) to its customers, so as to provide some assurance to the users that any critical QoS requirements will be met.

For single-user applications, each individual user may choose to determine their appropriate level of QoS and select a QoS class or level accordingly. This is generally expected to be the case, even if the individual users are related, for instance if they are using a common application. Although the early multicast QoS work required that all members of a multicast group receive the same level of QoS, layered multicast techniques, such as Receiver-driven Layered Multicast (RLM) [136] or Receiver-driven Layered Congestion control (RLC) [173], evolved to allow individual group members to choose their own level of QoS. These mechanisms are designed for heterogeneous network *conditions*, rather than heterogeneous user *preferences* — it is assumed that a group member will allow an application to use as much of the network resource as possible.

Assuming that users act according to their own individual preferences is common in the social sciences. Economists, for instance, assume that users, or agents, act in a **rational** manner. As formalised by von Neumann and Morgenstern [198], expected utility theory holds that peoples' preferences for potential expected outcomes can be ordered in a **utility function**. Rational man is out to maximise his individual utility, and given the choice between two outcomes, will choose the one which gives them the most benefit. Members of a group, therefore, would be unlikely to consider the other members of the group when choosing their level of QoS, since each member would choose the level which satisfies them the most.

How realistic is the assumption of a purely utility-maximising user? Is a member of a group going to be completely selfish, ignoring the presence of the other group members, grabbing as much of the finite network resource as possible, perhaps even at the expense of these other members? There are several reasons to be wary of accepting the rationality assumption at face value. Firstly, one theory might not necessarily be able to explain all that we observe in a real-world setting. As John Milnor states:

> "no simple mathematical theory can provide a complete answer, since the psychology of the players and the mechanism of their interaction may be crucial to a more precise understanding" [144].

Although Milnor is referring to the players in a game-theoretic system, the same holds for any human interaction. It might therefore be beneficial to examine (networked) game players in a real-world environment, to see whether they do in fact act in accordance with expected utility theory.

There is also reason to believe that users in a group environment might act differently to a set of completely independent users, irrespective of differences between theory and the real world. Users of a network, or users of a certain product who can be seen as forming a network, gain mutual benefit from sharing the network. An additional user joining the network increases the value for all users. For instance, a telephone network is useless unless there are other telephone users to which one can make a telephone call. Each additional telephone user adds to the usefulness of the network for all of the other users. Indeed, unless the network reaches a sufficient number of users, that is, a "critical mass point", the network might never reach a stable equilibrium and instead collapse [61].

The thesis that is being examined here is that users prefer a group level of delay, rather than simply choosing to maximise their own performance, for instance by always choosing the server which has the lowest latency between the server and their client. The enjoyment, or the utility, that a game player receives, is therefore dependent on the network conditions that the other members of the group are experiencing.

It is well-known that the value of a group activity to an individual participant may be related to the number of participants in that group. This has been informally described by engineers as Metcalfe's Law (the value of a network is proportional to n^2, where n is the number of users [141]), or more recently by David Reed as the Group-Forming Law (the value of the Internet is proportional to 2^n [170]). Economists, however, generally refer to these effects as positive consumption or **network externalities** [155]. For example, Katz and Shapiro define network externalities as:

> "products for which the utility that a user derives from consumption of the good increases with the number of other agents consuming the good." [110]

Network externalities have most commonly been studied in terms of standardisation and compatibility, e.g., the take-up and acceptance of fax machines [61], or the video games console market [74]. They are not limited to closed networks, and Henriet and Moulin [89] present a cost allocation scheme for open networks where users share costs according to the network externalities that are accrued. Social networks may also exhibit network externalities, for instance in the academic community [66].

One might expect that multiplayer games exhibit network externalities. The purpose of a networked multiplayer game is to participate with other people; if a user wishes to play against electronic opponents there would be less need for the networked aspect of the game (unless, for example, a user wished to play against a far more powerful computer such as the chess matches between Garry Kasparov and IBM supercomputers [94]). In general, however, it is reasonable to assume that a given participant in a networked game is taking part because they wish to interact with other remote, human users, and therefore, that their utility is derived, to some extent, from the existence and number of these other users.

Network externalities can explain why users choose to play games in a group, since the value of the game is partly dependent on the existence of the other players. They do not, however, explain why a user might consider the preferences and conditions of the other users. Given the choice between a set of servers, will the rational game player always choose the server to which they are best-connected, or will they look at the other players and perhaps choose what is seemingly a suboptimal server, in terms of network conditions?

An economic explanation for why users may seem to behave irrationally towards others is not a new concept. In 1759 Adam Smith published *The Theory of the Moral Sentiments*, which begins:

> "How selfish soever man may be supposed, there are evidently some principles in his nature, which interest him in the fortune of others, and render their happiness necessary to him, though he derives nothing from it except the pleasure of seeing it." [187]

In 1950 Leibenstein [121] described "bandwagon effects", where the demand for a good may increase because people copy each other in consuming a good, as new consumers desire to be associated with the group of original consumers. Perhaps game players might prefer to play on popular game servers, despite inferior network conditions, because they are following the crowd?

On the other hand, Postelwaite [162] suggests that the desire to have a favourable position in relation to others is part of human genetic programming, since animals similar to humans, such as apes and chimpanzees, have hierarchical social structures, where the top-ranked members receive favourable treatment compared with those lower down the scale. Those who jostle for a higher relative ranking are more likely to survive, and so natural selection means that we may have evolved to be concerned with our position in a group. If so, game players might prefer better-connected servers, whereby they can be the best-connected in the group.

Both of these considerations are part of what economists refer to as **interdependent preferences**. Interdependent preferences were first examined in 1949 by Duesenberry [59]. He recognised that a "real understanding of the problem of consumer behavior must begin with a full recognition of the social character of consumption patterns". Classical theories about rational consumers can describe human desires, but cannot explain how these desires come about. For Duesenberry, self-esteem and social status lead consumers to look at the preferences at consumption patterns of others.

Since the initial publication of Duesenberry's thesis, interdependent preferences have been observed by several economists in practice, and have been used to explain why people give to charity [9], to help determine the number of hours that people in a group are willing to work [28, 13], and to explain why people might be willing to spend money to ensure that all the members of a group receive an equal payout [205].

If we accept that group dynamics might affect the behaviour of users in a group, such that they behave differently to a set of independent users, then the existence of interdependent preferences means that group applications have an additional QoS parameter to consider — the **variation in a QoS parameter between the members of a group**. For example, if one user in a group is receiving 90% packet loss, this can affect both the afflicted user and the rest of the group, since neither can communicate with the other.

2.3 Networking techniques

We have discussed the importance of delay in multimedia applications such as games. In order to understand how networked delay might affect a game player's experience, it is useful to understand the causes of delay, and how games attempt to deal with the existence of this delay.

The causes of delay in CSCW applications are examined by Ingvaldsen *et al.* [95]. These can be divided into **end-system** factors, such as video compression and decompression, and packetisation of data, and **network** factors, such as the actual propagation of data packets through the network. The network is found to be the primary cause of delay.

2.3.1 Network architecture

Many of the networking techniques utilised in games are originally derived from the DIS standards for military simulations [96]. In a military scenario, all the clients are trusted, and so they can report their position and movements to all the other clients in the simulation. Thus, fully distributed peer-to-peer architectures are feasible and are commonly-used. In most games, however, clients are not trustworthy, and the incidence of cheating in networked games is very high. If a client were solely responsible for reporting its location, it would be easy to manipulate

the client in order to fake this information. Most games, then, have chosen to use a client-server architecture[2], and the server acts as an authoritative source of information and maintains a consistent state for the players. The clients are trusted with as little information as possible, and in some games, are "nothing more than a way for the user input to be sampled and forwarded to the server for execution" [24].

As well as reducing cheating, the other benefits of a client-server architecture are outlined by Funkhouser [73]. The message distribution function is moved out of the clients and into the server, which means less load on the client machines. The application is able to scale better, since processing, storage and bandwidth requirements scale with the density of entities, rather than the total number of entities as in a peer-to-peer environment. The main drawback of a client-server topology, however, is that there is increased latency. Blow [29] outlines the additional delay created by a client-server game. Delay at the client comprises **observation lag** (the rendering of the world and its comprehension by the player), and **influence lag** (the player's input and responses). A server has to receive messages, process them, and then retransmit back to all of the players. This is potentially true even for updating the player's own moves, since only the server has the authoritative copy of the world's database, and so a player may need to wait for an updated copy of this database before they can be sure of their own whereabouts.

2.3.2 Synchronisation delay

In the game *MiMaze* [124], a **synchronisation delay** mechanism is used to compensate for the differing delays experienced by different players. A global clock for all the players is provided using NTP (Network Time Protocol). Time is divided into sampling periods, and state update packets are timestamped and placed in buckets, associated with each sampling period. Each client only considers packets which are in the "current" bucket. All the clients therefore synchronise to a common clock, irrespective of what their actual network latencies might be. In *MiMaze* the synchronisation delay is set to 150ms, such that a packet issued at time t is actually processed by the client in the bucket containing t+ 150ms. If, however, the network delay is in excess of 150ms, the update packets would be discarded.

2.3.3 Dead reckoning

Players in a client-server game are dependent on receiving state updates from the server in order to know what is happening in the shared virtual world. It would be extremely expensive, both in network and computational resources, to send updates on a continuous basis, and so state updates are sent at time intervals. The clients therefore need a method of approximating the

[2]Confusingly, some games, e.g. *Age of Empires* [192], describe themselves as being peer-to-peer, when in fact they are client-server, with one client acting as a non-dedicated server.

state of the world in between receiving these updates, or in the event of packet loss. One of the most commonly used methods is **dead reckoning** [77, 11]. Using the previously-established location information for an object, the movement of the object can be predicted.

Several methods for prediction are outlined in [157], such as predicting by assuming constant velocity or acceleration, predicting the position of an object, or predicting the input of a player. The optimal prediction method can depend on the type of game, and the type of input device that is likely to be used — for instance, a driving game might require an input device with different characteristics to the input device for a shooting game, and a games designer could optimise the prediction scheme accordingly.

As with the choice of network architecture, cheating is a concern for a dead reckoning implementation. Baughman and Levine [20] describe a possible method for cheating under dead reckoning by deliberately dropping state updates. If a dead reckoning policy allows n state updates to be approximated before the player is assumed to have disconnected, then a player could deliberately drop $n-1$ updates without being detected, as long as the nth update is sent. The other players will then be forced to approximate the player's position, but will only be able to confirm the position every n updates. Depending on the value of n, and the accuracy of the prediction algorithm, this might make it possible to cheat. Baughman and Levine propose a *lockstep* protocol, whereby all the players' clocks advance synchronously. Lockstep can prevent cheating, but at the expense of having to disable dead reckoning. An anti-cheating mechanism that can operate under dead reckoning is proposed by Cronin *et al.* [49]. Rather than wait for each player to send the details of their move to each of the other players before the game clock is advanced, each player can send a number of moves at once in a *pipeline*. Using a pipeline, however, means that a player could wait for another player to send a set of pipelined moves before responding, and thus potentially benefit by knowing all the other player's moves. To prevent this "late-commit" cheat, an adaptive pipeline can be used, where all the players measure their delay to the other players, and adjust their pipeline sizes accordingly. While the pipeline size is being adjusted, moves are placed into a secure buffer, to prevent exploits.

2.3.4 Consistency

Dead reckoning can help to minimise the effects of latency, but it can also introduce problems of its own. Consider player A attempting to shoot player B in an FPS game (Figure 2.5(a)). Player A fires their gun, and dead reckoning shows A that B is now dead (Figure 2.5(b)). There is 200ms latency, however, between A and B, and before B receives the indication that they have been shot by A, B shoots C (Figure 2.5(e)). Should C be dead (Figure 2.5(f)), since they

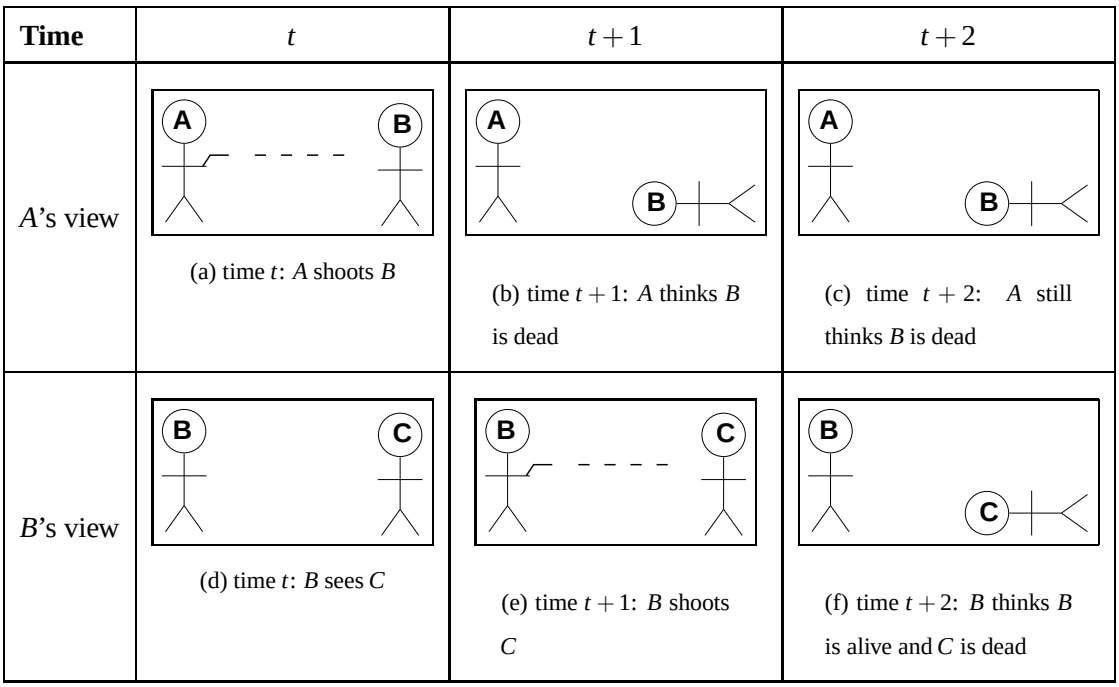


Figure 2.5: Inconsistency in gameplay caused by network delay

were shot by *B*, who was already dead (Figure 2.5(c))? Network delay can therefore lead to inconsistencies between each user's state.

One method for resolving these inconsistencies is a timewarp algorithm [134]. The application periodically makes snapshots of all the user's state. Whenever an inconsistency is detected, a "timewarp" is performed, reverting the state to the last recorded snapshot. Timewarping has been implemented in the game *Half-Life*, where it is named "lag compensation" [24]. Keeping snapshots can be memory and computationally expensive, however, and some researchers have proposed using multiple servers instead [50].

Although rolling state back to a previous snapshot can reduce inconsistency, it also makes for a disjointed experience for the user. To return to the previous example, player *C* would now die, and then be brought back to life after the timewarp occurs.

2.4 Discussion

Networked games are becoming an increasingly popular application on the Internet. Although in some respects, games are not particularly interesting from a research point of view, as they are simply commercial implementations of applications developed in the research community, such as NVEs and DIS, we believe that their sheer popularity merits closer examination. A research NVE will never have as many users as a game, since only a small proportion of computer users have access to research and academic networks. Therefore, we claim that the study of

networked games will be useful to further research in NVEs, since they provide a guide to how users will behave in large-scale NVEs.

There are also important differences between games and research applications. Military simulations such as DIS are generally designed for purpose-built networks, or at least networks which can provide a guaranteed level of QoS. Reliability is a primary concern, and the DIS standard stipulates that 98% of PDUs be delivered. This is a result of the nature of the task — in a military situation, it is vitally important to know exactly the location of all your vehicles, and which of your targets have been destroyed. Reliability is emphasised over other attributes of the simulation, such as the realism of the graphical representation. It could be argued that in a game, the graphics are more important than reliability. The packaging and advertising for any commercially-available game will undoubtedly advertise the realism of the graphics, the advanced 3-D features, and so on, rather than the reliability of the network protocols. This may also be due to the nature of the task — the consequences of an error in a game are unlikely to be as serious as those of a miscalculated miltary manouever. This is the fundamental difference between a game and a simulation — "A simulation is a serious attempt to accurately represent a real phenomenon in another, more malleable form. A game is an artistically simplified representation of a phenomenon" [47]. On the other hand, the relationship between military applications and games is an important one, and Lenoir argues that in the future it may be difficult to distinguish between the two [123].

Network latency appears to be the primary concern of most users of networked games, and has also been highlighted by the research community as a particular problem. In addition to latency, we must consider the interaction between the members of a group application, and how this might affect preferences for QoS.

We have discussed how latency in a game can lead to several effects which are observable by the end users. Consistency, the possibility of cheating, jerkiness and disjointed state updates are just a few of these. Game developers have designed methods such as dead reckoning to cope with the effects of network latency, but these are not perfect, and can create additional problems. Whilst this thesis will not concentrate on these latency-compensating techniques, we can determine from our discussion that we would expect a player to desire a minimal amount of network latency between themselves and the game server and other players, and to be able to notice the effects of latency.

2.5 Summary

In this chapter we have discussed how networked games have evolved from the relatively simplistic *Tennis for Two* in the 1950s to the popular FPS, RTS and MMORPGs of today. We have noted the following points:

- Games are closely related to NVEs, DIS and CSCW applications
- Games may be more impressive graphically, but less reliable than military simulations
- Bandwidth is not a large problem for current games
- Latency is the most important QoS parameter
- Games use a variety of techniques to deal with latency, e.g., dead reckoning or synchronisation delay
- Multiuser applications may exhibit network externalities or interdependent preferences

In the next chapter, we will discuss some of the work which is currently being carried out to address these problems. We will then outline the area which this dissertation will examine.

Chapter 3

Group preferences and Quality of Service for games

In Chapter 2 we looked at the evolution of networked games, and we have seen that multiplayer games are becoming a popular networked application. We have claimed that due to the popularity of these games, they are an important application to study, so as to aid understanding about research into networked applications and NVEs, and that it is also worthwhile to study games in their own right. Network delay was noted to be a particular problem, and one that users tend to consider when playing networked games. We have also discussed how the interaction between users in a group might affect behaviour and user preferences for QoS, and suggested that this needs to be examined in more detail.

In this chapter we survey some of the work that is most closely related to this area. We then explain the focus of this dissertation, the game that we have chosen to examine, and the reasoning behind this choice.

3.1 Related work

3.1.1 Network analysis

Several researchers have conducted network-level analyses of games. Borella [31] analyses the FPS game *Quake*. Packet-level network traces are taken on a local area network and analysed. The game server is found to send packets in bursts, and each burst consists of packets being sent on a nearly continuous basis, with very small interarrival times. The interarrival time between the bursts can be modelled by a split extreme distribution — the lower 50% is deterministic, whilst the upper 50% is exponentially distributed. In addition, the interarrival distribution is very heavy-tailed, and exhibits a high level of autocorrelation. Different clients receive different amounts of data; this is to be expected since different players will be in different parts of the virtual world and thus require different update information. Färber [65] compares Borella's

results with those for the game *Half-Life*. As explained in Chapter 2, *Half-Life* is based on *Quake*, and so it is perhaps unsurprising that the results are similar.

Bangun *et al.* [17] look at both *Quake* and the peer-to-peer game *Starcraft*. *Starcraft* differs from *Quake* in that packet interarrival times decrease as the number of players increase — this is probably due to its peer-to-peer architecture. In [16] Bangun *et al.* also look at the client-side packet size distribution of *Tribes* players. *Tribes* is a team-based FPS game which also uses a client-server architecture with UDP network messages. The packet size distribution is found to be heavy-tailed, with 90% of the packets smaller than 60 bytes.

Joyce [109] also looks at FPS games, taking measurements of the games *Quake* and *Unreal*, but rather than examining a local area network, analyses games on a WAN by looking at the traffic traversing a large ISP peering point. Packets sent by the clients are generally small (50-70 bytes), whilst the server sends packets in the 50-150 byte range. Both games, despite being designed by different companies, are found to have very similar networking characteristics.

These studies only look at activity at the network level. To understand why an application is creating particular patterns in network traffic, it can be useful to examine activity at the application, or session, level. Greenhalgh *et al.* look at network and session-level behaviour in an interactive television program [80]. They find that the application is characterised by bursts of coordinated activity, and there is little sense of "turn-taking". Almeroth and Ammar [6] look at session-level behaviour in multicast single-source video sessions, and find that users' session duration is exponentially-distributed. Both of these findings could make provisioning for group applications more difficult, since the bursts may occur when the application is more interesting for the users, and so network congestion or loss at these points would be most disruptive.

3.1.2 Network performance, congestion control and QoS

Fischer *et al.* [68] note that it may be optimal from a network point of view if members in a group application do coordinate their preferences. If there are three receivers $\{r_1, r_2, r_3\}$ of a multimedia stream downstream of a bottleneck, and $\{r_1, r_2\}$ are receiving high-quality video, whilst r_3 is receiving low-quality, then in the absence of a layered encoding scheme, it would make sense for r_3 to receive the high-quality stream as well, or for $\{r_1, r_2\}$ to change to the lower-quality stream. A system of agents is proposed, which interact with each other, negotiating and renegotiating QoS requirements between multiple applications and users, in such a manner that the overall QoS and utility of all the users may be maximised. Kouvelas *et al.* [114] present a scheme whereby receivers can self-organise into groups to maximise the quality of their received streams.

QoS for delay-sensitive applications is a topic that has been addressed by many researchers. Both the IETF IntServ and DiffServ standards deal with delay-sensitive applications: RFC1633 [33] states that the "core service model is concerned almost exclusively with the time-of-delivery of packets", whilst the DiffServ standards include an Expedited Forwarding Per-Hop Forwarding Behaviour (PHB) for delay-sensitive applications [107]. Key *et al.* [112] present a set of packet-marking schemes for delay-sensitive applications, and propose that these schemes may be useful, depending on users' sensitivity to delay.

Probabilistic Congestion Control (PCC) [200] is a congestion control scheme designed specifically for games and similar applications. Games may have a strict lower bound on the amount of traffic that is required for gameplay to make sense — if a player is only receiving a state update once a minute because of a bottleneck link, their game is likely to be impaired to the extent of being unplayable. Rather than implementing congestion control on each individual flow, PCC proposes that all of the players in a game should be jointly considered. Thus, in the event of congestion, rather than reducing the traffic sent to and from individual player, which might make the game unplayable for everyone, one or more flows could be terminated, i.e., one or more players could be dropped from the game. If the players of a game share a common bottleneck, it is possible to use PCC to keep the sum of the flows generated by all the game players TCP-friendly, and yet allow the other players to carry on playing unimpeded, which might be preferred by game players.

One research area that has considered the preferences of a group in a group application, rather than the preferences of the individual members of the group, is multicast congestion control. Proportional fairness has become a popular metric for allocating bandwidth between flows on a congested link [111]. This relies on the assumption of individual users having logarithmic utility functions. It is unclear, however, whether this same logarithmic utility function should be assumed for a multicast or multipoint transmission. Should a multicast flow, which may represent the usage of several users, receive a larger bandwidth share than a unicast flow? To limit a multicast flow means that a larger number of users will be penalised. On the other hand, by using multicast, those group members may be receiving better network performance than if they had all chosen to conduct simultaneous unicast communications. Chiu [43] shows that proportional fairness may produce an "unfair" outcome in the multicast case, and suggests a weighted proportionally fair solution, where multicast flows receive a bandwidth share weighted according to the aggregate utility of the downstream receivers. Legout *et al.* [120] suggest three possible strategies for allocating the bandwidth amongst the downstream receivers. A Receiver Independent strategy is where bandwidth is allocated equally amongst the downstream multi-

cast and unicast users, so that multicast users are treated the same as unicast users. The Linear Receiver Dependent (LinRD) stategy determines the bandwidth share of a multicast stream according to a linear relationship with the number of downstream receivers, i.e., the multicast stream receives the aggregate bandwidth that the receivers would have gained if they had each used unicast. Finally, a Logarithmic Receiver Dependent (LogRD) strategy attempts to reward users for choosing multicast, by increasing the bandwidth share to a multicast stream logarithmically as new receivers join. Whilst these strategies are designed for multicast applications, they might also apply to multiuser applications such as games.

The commonly-used client-server network architecture has several drawbacks from a networking perspective. A bottleneck that is near a central server will impair performance for all of the players. Several researchers have recently proposed proxy mechanisms, whereby a set of servers can be distributed to various hosts [44, 135, 19, 82]. A proxy system can aid robustness by providing alternative routes and servers, aid congestion control, and help to prevent cheating by timestamping actions and limiting the amount of information available to clients. Min *et al.* use knowledge about the players' respective locations within the game to carry out load-balancing across a group of servers [145]. Tran *et al.* use a Java-based middleware approach to achieve a similar goal [194] — the virtual world simulation is replicated across a number of simulations, each of which has a "master" replica, and a number of "slave" simulations, which can be viewed as proxies.

3.1.3 Delay requirements

The delay bound for real-time multimedia applications, that is, the level of delay above which performance becomes impaired, has been studied by researchers in a variety of fields. Human factors research indicates that a round-trip time of 200ms might be an appropriate limit for real-time interaction [15]. The IEEE DIS standard stipulates a latency bound of between 100ms and 300ms for military simulations [96]. MacKenzie and Ware find that in a VR (Virtual Reality) environment, interaction becomes very difficult above a delay of 225ms [129]. The ITU G.114 standard recommends between 0 and 150ms for a one-way transmission time for voice communications, although up to 400ms is considered acceptable [102]. Park and Kenyon [158] examine a two-user cooperative task in an NVE. Performance with 200ms latency is significantly worse than 10ms, and jitter is also found to have a significant effect.

There have been few studies of commercially-available networked games in particular. Pantel and Wolf [156] examine two car-racing games, and find that "a delay of more than 100ms should be avoided", although they do note that different types of games might have differing requirements. For instance, Schaefer *et al.* [177] examine players of another type of game,

the shooting game *XBlast*, using a Mean Opinion Score (MOS) methodology, and find that a delay of 139ms is acceptable. Vaghi *et al.* [197] analyse the effects of delay in a simple ball game implemented on the MASSIVE NVE. They find that delay becomes perceptible through discontinuities and visual anomalies in the game. Apart from these studies, many of the delay requirements for games have been extrapolated from those for other real-time applications. Cheshire [42] proposes a latency bound of 100ms for networked games, although no empirical basis is given for this.

Common to all these studies is that typically very small groups or single-user tasks are studied, and all users are assumed to share similar network characteristics.

3.1.4 Behaviour of game players

To observe and understand user behaviour in networked games requires some sociological analysis of the players and the games themselves. Sociologists have been studying the effects of games since at least the early 1980s [163]. This research tends to concentrate on the content of games, such as violence or narrative in video games, which is perhaps not very interesting from a networking perspective. For instance, several researchers look at aggressive behaviour by players of computer games [58].

User behaviour in games may differ from that in other multimedia applications due to the nature of the application. Game players can exhibit symptoms of addiction [34, 81], and online gaming is considered by some researchers as an example of pathological Internet use [147]. It has also been shown that users react differently when playing against other people as opposed to computer-generated opponents [202].

Manninen [131] looks at the interaction between users in games, in particular *Counter-Strike*, which is a variant of *Half-Life*. Several methods of interaction are found to be present, such as the appearance of a player's avatar, gestures, physical contact (within the context of the gaming world) pre-programmed moves, and modifying the virtual world. This study takes place on a LAN, and we might expect results to differ from actions on the public Internet: "the LAN gaming sessions indicate the need for strong social togetherness, and thus, are often the venues for the strongest experiences" [132]. Players have to make a special effort to attend such "LAN parties", often bringing their own equipment with them, and thus their tolerances might differ from a spontaneous Internet gaming session. Another behavioural analysis of a computer game can be found in [108], but again this is in a non-networked environment.

3.1.5 Other games research

We have discussed how games are worthy of research by virtue of their popularity. Since such games are popular, it makes sense to use them for understanding user behaviour, rather than writing a custom application for research purposes. This aspect of games has been noted by many other researchers, who have also chosen to leverage existing games [126]. For instance, FPS games have been used to study "e-learning" and the visualisation of knowledge spaces [72, 105], context-aware services [36] and Artificial Intelligence (AI) [2, 116].

3.1.6 Discussion

Games are becoming an increasingly popular area of research, and we have outlined some of the related work. There have been several network-level analyses of games, but little corresponding analysis at the session level. Resource management schemes for networked games and multiuser applications have been proposed, but these have not been examined with user behaviour to see if they would be acceptable to game players. The delay requirements for multimedia applications are well-known, but there has been little work examining multiplayer games in detail. Sociological studies of games have concentrated on aspects which are not specific to the networked game, such as the narrative, although the interaction between players in games indicates that the social and group effects discussed in the previous chapter might be present.

We thus note several areas which need to be examined:

- Session-level user behaviour of networked multiplayer game players
- Delay requirements for multiplayer real-time networked games
- QoS preferences for groups of users in multiplayer games

3.2 Approach of this dissertation

As games are one of the most popular NVEs, and are played by millions of users across the Internet every day, a potentially useful method of learning about user behaviour in games is to examine the actions of these current game players.

We have chosen to concentrate on the FPS game. The most popular MMORPGs generally operate a small number of servers, and monitoring user behaviour would require access to these servers. Gaining such access is difficult, because many of the companies that run such servers do so on a commercial basis and are reluctant to divulge usage data or allow third-party monitoring. The server program for these commercial MMORPGs is not made available, and so it is not possible to run an alternative server. A few open source RTS games and MMORPGs

Average number of servers	**Average number of players per server**	**Maximum number of players on a server**
25.50	0.563	30

Table 3.1: Average number of *FreeCiv* players

exist, such as *FreeCiv* [71], but the number of players that play these opensource games is too low to make any monitoring worthwhile. We polled the *FreeCiv* "metaserver" (which provides a list of all the currently-available *FreeCiv* servers) every six hours for four months, the results of which are shown in Table 3.1. The maximum number of players observed on a single server was 30, and servers were usually empty. This is far lower than the hundreds of thousands of players who regularly play commercial MMORPGs like *Everquest* and *Diablo*. It is unlikely that the interactions between such a small number of users on a *FreeCiv* server would be representative of the interactions on their commercial counterparts.

In comparison, FPS games comprise thousands of servers, each of which services a small number of players. Anyone can run one of these servers, as the server daemon program is typically provided with the client software. Running a server negates the need to rely on other server operators, since usage data on the players can be gathered locally.

Another reason to choose the FPS game over the RTS game or MMORPG is that FPS games are generally thought to be more delay-sensitive than RTS games or MMORPGs. Terrano and Bettner describe testing carried out during the development of the RTS game *Age of Empires*: "250 milliseconds of command latency was not even noticed - between 250 and 500 msec was very playable, and beyond 500 it started to be noticeable" [192].

Of the FPS games, we have chosen to study the game *Half-Life*, a game released by Valve Software in 1998. Valve licensed the *Quake/Quake II* graphics engine and networking code from id Software, and so from a networking perspective, *Half-Life* is very similar to the *Quake* family of FPS games, as well as the newest generation of id Software FPS games such as *Return to Castle Wolfenstein*. These games represent the majority of FPS games currently played on the Internet, and in any case it has been shown that FPS games not based on this code also exhibit similar networking characteristics [109].

At the time that our studies were carried out, *Half-Life* was by far the most popular FPS game. Figure 3.1 shows the average number of servers available on the Internet for the most popular FPS games. These figures were obtained by polling the master servers for each of the

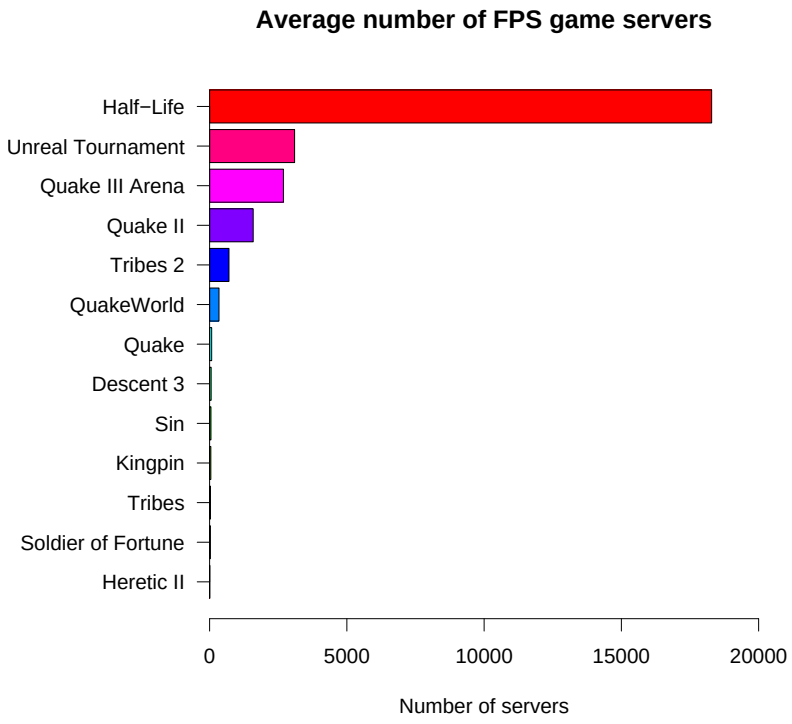


Figure 3.1: Average number of servers for different FPS games

games twice a day over the course of a year. As there can be thousands of servers available to a player at any given time, the master servers provide a mechanism for discovering these servers. A game server registers with the master server on startup, and the master server responds to queries from players by providing a list of active servers (more details about the master server mechanisms can be found in Appendix B). The number of servers provides an indication of the popularity of a game, and the number of *Half-Life* servers exceeds the total number of servers for all the other FPS games.

As new games are introduced, old games will wane in popularity. This is verified by McCreary and Claffy's study [138], which shows that over time, older games such as the original *Quake* generated less traffic than newer games such as one of its more recent successors, *Quake III*. Figure 3.2 plots the weekly average number of servers for each FPS game from the queries in Figure 3.1, over time. It can be seen that *Half-Life* was increasing in popularity over this time period, with approximately 15,000 servers in June 2001, rising to approximately 28,000 servers by September 2002 (the occasional drops in the number of servers are probably due to connectivity problems affecting the centralised master servers). In comparison, the number of servers

for the other FPS games have relatively static growth rates. *Half-Life* is thus an appropriate game to study, since it is both popular, and representative of most FPS games.

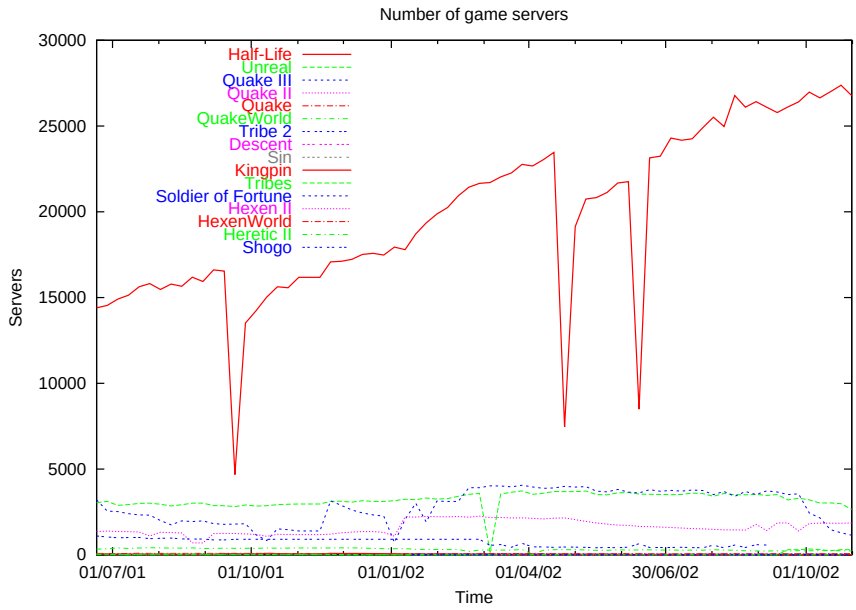


Figure 3.2: Number of game servers over time

3.3 Summary

In this chapter, we have outlined some of the current research that is taking place in the area of networked games, and have described the problem and area that this thesis will examine. We have noted the following points:

- Research has examined network-level statistics, congestion control and delay requirements for generic NVEs
- Areas that still need to be examined include examine session-level behaviour, delay requirements for games, and QoS preferences in group situations
- FPS games are easier to study than RTS games or MMORPGs due to server size and commercial considerations
- *Half-Life* is the most popular FPS game, and one that is representative of the genre

Chapter 4

Session-level join-leave behaviour in FPS games

4.1 Introduction

In the previous two chapters, we have discussed the nature of FPS games, and how playing such a game is a group activity. In a group activity, it is important to have a number of players in a game in order to create a worthwhile playing experience for the users. We have therefore speculated that playing an FPS game might exhibit network externalities, whereby the utility that a player receives from a game is related to the number of players in the game. In this chapter we attempt to demonstrate the presence of these network externality effects — that the existence of other players in a game affects a user's decision to join or leave a game. In order to show this, we directly observe game players who connect to publicly-available FPS game servers.

As we have already described, FPS game servers tend to run games for an indefinite period of time, and users are free to join and leave at any time. There are several thousand servers running on the Internet at any given time (see Figure 3.1). These servers are publicly accessible by anyone who is connected to the Internet. Through the game-specific query mechanism, it is possible to monitor these servers, and thus to gain an understanding of user behaviour in these FPS games.

4.2 Methodology

Almeroth and Ammar [6] monitor a number of IP multicast sessions by joining a session and then watching the other session members join and leave. This is impractical for networked games, however, since to join a game implies participation. Passive users are generally disconnected from the game if there is no activity for a certain period of time. Game servers also tend to have a fixed maximum number of players which can connect to the server — to take

up one of these slots with a monitoring program would probably incur the wrath of the players and server operator, and lead to the monitoring host in question being disconnected and banned. This means that a monitor would have to be a physical player. As most people are only capable of playing one game at a time, and only for a certain number of hours a day, this limits the scope of any data collection. Although it is possible to simulate a user through a script or program, such "bots" are also frowned upon by many game server operators and the gaming community, and the use of these scripts could lead to the user in question being barred from that server. Moreover, joining a server as an additional player might create problems, in that the hypothesis we are testing is that user behaviour depends on the number of players in a game. To test this by joining a server, thus altering the number of players, would affect the results. A **passive** monitoring system was required, such that the behaviour of players could be monitored without participating or interfering in the application.

Many game servers offer a query mechanism, whereby specific variables about game status can be retrieved. Since joining and continuously monitoring games seemed impractical, polling and querying game servers at regular intervals was determined to be the next best option. By polling servers and determining the number of players at each poll, an approximation of user behaviour can be obtained. Many networked games also allow the querying of such variables as players' nicknames and the amount of time that they have been playing, and so the duration of each users' session can also be estimated. The accuracy of this method depends on the frequency of polls. If the polls are spaced too far apart in time, then any users who join and leave between polls will be missed. If the polls are too frequent, the amount of network traffic might have an effect on the servers and perhaps affect user behaviour.

Data were collected using the *QStat* tool [165], which is a program designed to query and display the status of game servers. *QStat* supports a large number of online multiplayer games. We have already mentioned that *Half-Life* is one of the most popular FPS games. *Half-Life* was also one of the games which supports the reporting of a player's connection time, and this was another good reason for using *Half-Life* for our study.

A list of 2193 IP address/port pairs[1] of hosts running the *Half-Life* server daemon was obtained from a "master server" at `half-life.west.won.net`. The master server's list is composed from submissions by server administrators and/or automatic registration by servers (depending on the game). This list may also be queried by users through the application itself, or through the use of some of the aforementioned programs for determining the closest or quickest-

[1]It is not uncommon for a single machine to run several servers on different ports; of our list of 2193 servers, there were 1725 unique IP addresses.

responding game server. This list should therefore be a representative set of publicly-available game servers.

The servers were polled at regular intervals, as depicted in Figure 4.1. A single machine at University College London (UCL) was used to sequentially poll each of the servers in the list. At each poll, the number of players, their chosen nicknames and the number of seconds that each player had been connected were retrieved (some example output from the query tool is shown in Figure 4.2). Servers might occasionally fail to respond to a poll. This might be due to transient congestion, since all the queries use UDP and thus do not take advantage of the retransmission features of TCP, or because the host was overloaded. If this was the case, we did not retransmit, since we have noticed that in certain cases this can exacerbate the effects which led to the original timeout [86]. Instead, we assumed group membership to be the same as at the previous successful poll, if the next poll was successful. We assume that two consecutive unsuccessful polls indicates that the server has indeed terminated, and group membership is zero.

There are several limits to this methodology. Since polling took place at the application level, we could not detect such events as unsuccessful join attempts, as these do not register in the game. We were also limited in that polling takes place from a central machine at UCL, and so any network failures that existed solely between UCL and the game servers (but not between the game server and the players) would affect our results. The findings in this chapter are reported in [87].

	Servers	**Game**	**Frequency**	**Duration**
O-I	2193	*Half-Life*	30min	1 week
O-II	35	*Half-Life*	5min	3 days
O-III	22	*Quake*	5min	1 week
O-IV	3	*Half-Life*	5min	2 months
O-V	3	*Quake III Arena*	5min	2 months

Table 4.1: Observations taken of game servers

Several sets of observations were taken; the differences between these, and the labels that are used in this chapter to refer to them, are shown in Table 4.1. The first set O-I used the aforementioned master list of *Half-Life* servers. From this, the 35 most popular servers were selected for more detailed observation over one weekend in O-II.

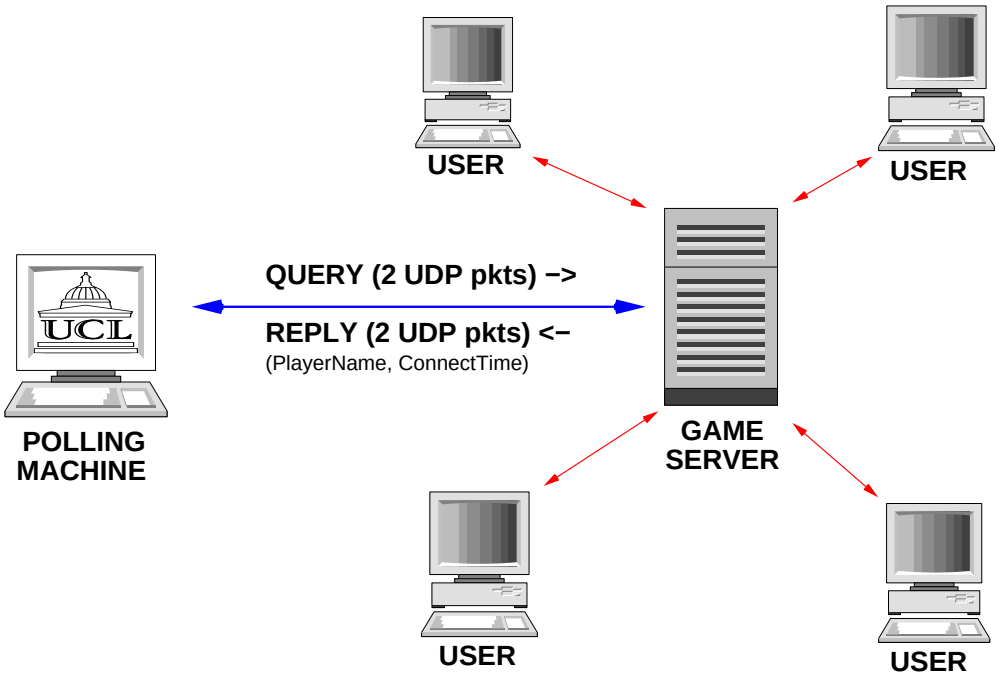


Figure 4.1: Data gathering setup

```
NAME: Merlin TIME:  5710
NAME: [F.u.T]The_LAW TIME:  5728
NAME: MagNETo [FH] TIME:  2176
NAME: TomiN TIME:  2409
NAME: [DEM] Guybrush T. TIME:  8575
NAME: [.HoF.]Ben Kenobi TIME:  1177
NAME: [Thug]Tosh TIME:  142
NAME: TDMT_Silvan TIME:  1540
NAME: Gulzak TIME:  874
NAME: [DBK]HannibalTC TIME:  954
NAME: [STANDARD] Kill Demon TIME:  1085
NAME: -=Phoenix=- TIME:  5593
```

Figure 4.2: Example *QStat* output

Set O-III used 22 *Quake* servers, the addresses of which were also obtained by querying a master server. *Quake* is an older game, introduced in 1996, which is why the number of servers is so much lower than *Half-Life*, which first went on sale in 1998. We chose to analyse *Quake*, however, because it is one of the few games to allow the querying of players' IP addresses, and that the sourcecode for *Quake* is freely available, and at the time we believed these aspects may have been useful for determining the network topologies and spatial analysis of games.

The last pair of observations, O-IV and O-V, come from two sets of servers which Microsoft Research have been running at their site in Cambridge, UK. Using a public list of servers

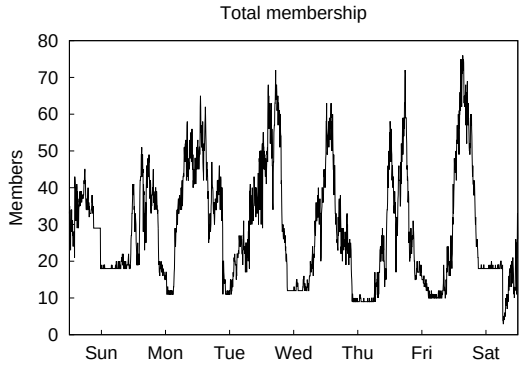
proved to have many difficulties. Some of the IP addresses which appear on the list of servers were dynamically allocated addresses (caused, for instance, by users running game servers via dial-up ISP accounts or other machines with intermittent connectivity). It would appear that the master server does not update its list frequently enough to eliminate these, and so many polls would end up targeting machines which were no longer running the game server. Of the original list of 2193 addresses, we found that 265 of these were never running the server during the course of our polls. The servers run by Microsoft Research had static IP addresses and thus we could be confident that they would be contactable for the duration of our polls. These servers consisted of two games, *Half-Life* and *Quake III Arena*. The game used in O-V, *Quake III Arena*, does not allow the querying of player duration. For this set of data we assume that each player joined at the time of the poll at which they are first noticed; this figure thus has a potential inaccuracy of up to two poll periods, that is, 10 minutes.

4.2.1 Summary of observations

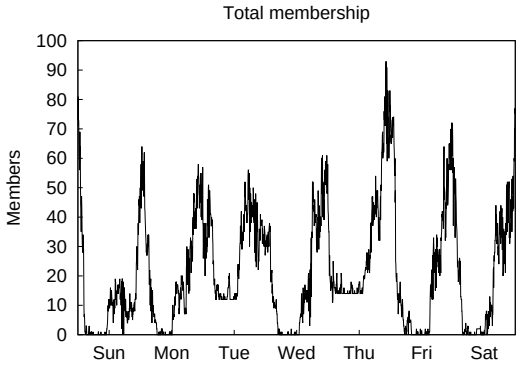
We observed a total of 1,757,539 individual sessions (i.e., individual users joining and then leaving a game server). Table 4.2 shows some of the overall aspects of the data. We were interested in examining three specific features: the number of participants in a game, the interarrival time between participants, and how long a player remained in a game.

	Total joins	**Average joins/server/hr**	**Median interarrival (sec)**
O-I	1510445	4.65	225
O-II	69961	27.76	70
O-III	37037	10.01	118
O-IV	23559	4.19	115
O-V	5872	1.04	300
	Max interarrival (sec)	**Median duration (sec)**	**Max duration (sec)**
O-I	246171	1576	3165999
O-II	17309	1098	66738
O-III	51706	618	410699
O-IV	77843	612	2614737
O-V	76800	901	403201

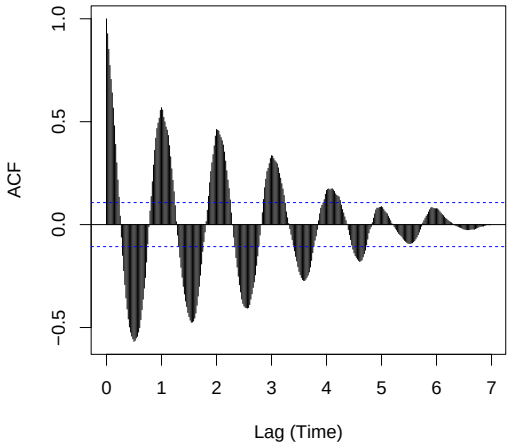
Table 4.2: Summary of session-level analysis



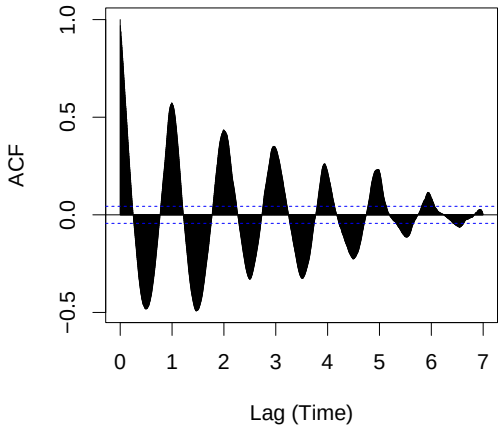
(a) single server from O-IV



(b) single server from O-IV



(c) Autocorrelation function of Figure 4.3(a)



(d) Autocorrelation function of Figure 4.3(a)

Figure 4.3: Number of users

4.3 Session membership

Figures 4.3(a) and 4.3(b) show the total number of players for two game servers, scaled to a one week period. The time axis in Figures 4.3(a) and 4.3(b) is set to what we believe to be the server's local timezone, having approximated the server's location by performing a whois on the server's IP address. It can be seen that the number of participants in a game exhibits strong time-of-day effects, peaking in the middle of the day. The strong sinusoidal pattern in the correlograms in Figures 4.3(c) and 4.3(d) also indicates seasonal variation.

Since the time-of-day effect is so clearly evident, it is possible to do a simple seasonal decomposition by subtracting each observation from the mean value for all the observations

taken at that time of day [40]. The results of this are shown in Figure 4.4, where the higher solid line represents the time-of-day effect, the lower solid line the remainder, and the dashed line the observed data. Three days are higher than the others; these, as one might expect, are Friday to Sunday. Game players evidently consider games a recreation, and as such spend their weekends participating in networked games.

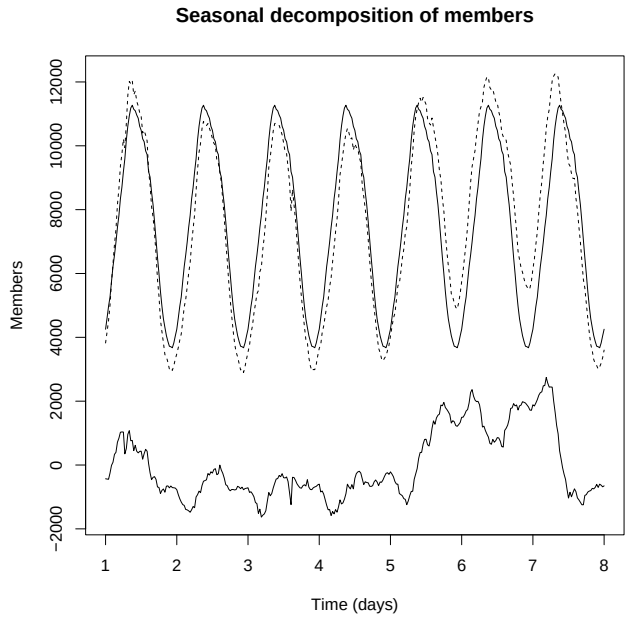


Figure 4.4: Seasonal decomposition of smoothed membership data

4.3.1 Network externalities

Figure 4.5 shows the temporal autocorrelation function (ACF) of the corrected data from Figure 4.4, after removing the time-of-day effect; this shows the degree to which the number of players in a subsequent time period depends on the session membership in the previous period. It can be seen that the level of autocorrelation is high, even for a large number of time periods. Thus, as expected, there appear to be some network externality effects.

Having observed the time-of-day and network externality effects, we analysed the session membership data using time-series analysis. We applied several ARIMA (Autoregressive Integrated Moving Average) models to the data (see Appendix A for a description of ARIMA modelling).

Figures 4.6 and 4.8 show the diagnostics for a $(1,1,1) \times (0,1,1)_{48}$ ARIMA model. In each figure, the top left chart indicates the residuals from the model, the middle left chart is the ACF of the residuals, and the bottom left chart shows the Ljung-Box statistic (see Appendix A.1). The right side of the figure shows a cumulative periodogram for the model, and is used to

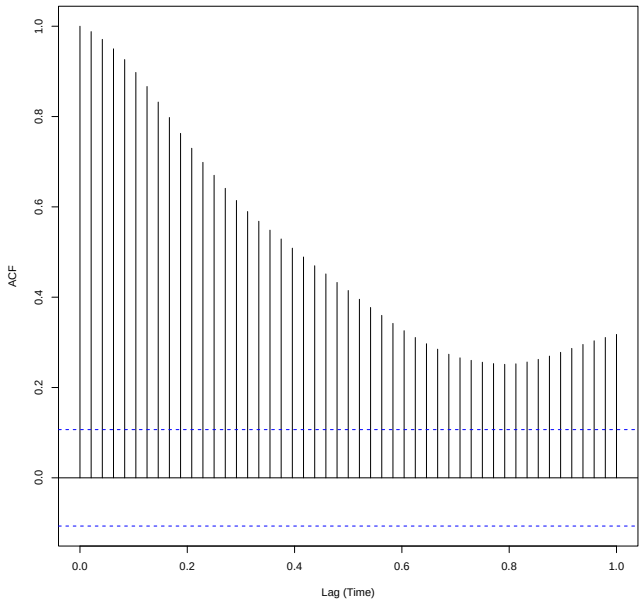


Figure 4.5: Temporal autocorrelation in number of players

indicate sinusoids (periodicity) in the model. Figures 4.7 and 4.9 show the diagnostics for a $(2,1,1) \times (0,1,1)_{48}$ model for the same pair of servers. The Ljung-Box statistic in Figures 4.7 and 4.9 indicate a higher goodness of fit.

A $(2,1,1) \times (0,1,1)_{48}$ model incorporates both network externalities, since the autoregressive component means that the number of players up to an hour prior to a player joining has an effect on a player's decision to join, and also includes the time-of-day effect through the seasonal $(0,1,1)_{48}$ component.

4.4 User duration

Game servers tend to run continuously, with users joining and leaving as they wish. As such it is not meaningful to discuss the overall session duration for the server, i.e., a whole game, since this can be construed as being the length of time that the server is running. Instead we examine the duration of each individual session, i.e., a user's game. We define a session as being the time between a player connecting to the server, and disconnecting from the server. A player being killed in the game and "respawning" (coming back to life) elsewhere in the virtual world does not constitute a new session. Figure 4.10 shows the duration of users' individual sessions from O-II: it can be seen that these durations vary quite widely, and that many game durations are lower than our polling period of five minutes. This might be due to dropped connections, or users browsing games by starting a session to see what is going on and deciding that a

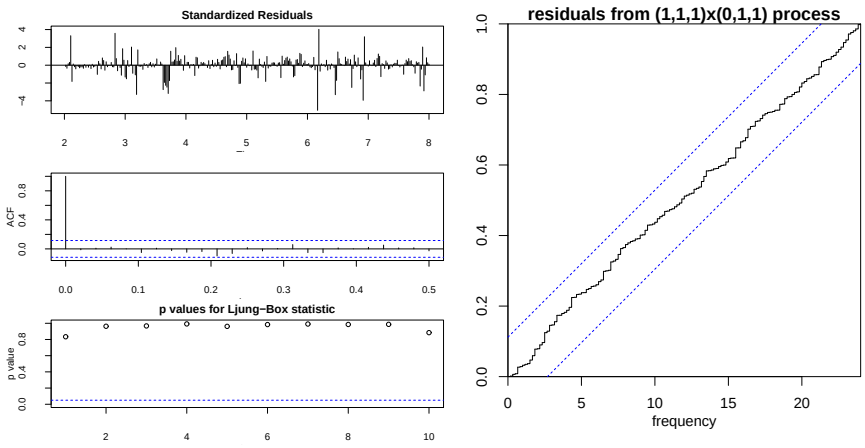


Figure 4.6: ARIMA diagnostics and cumulative periodogram for $(1,1,1) \times (0,1,1)_{48}$ model for a single server

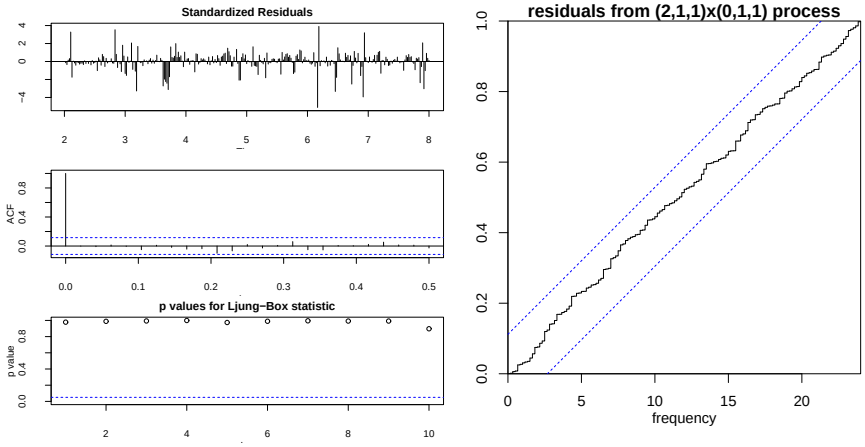


Figure 4.7: ARIMA diagnostics and cumulative periodogram for $(2,1,1) \times (0,1,1)_{48}$ model for same single server as Figure 4.6

particular game is not to their liking. At the other end of the spectrum, there are several long game durations of over 24 hours. These might be "hardcore" gamers, automated players/bots or users who have mistakenly left their connections active.

In Figure 4.11 we fit the user duration data for two individual servers to a set of randomly generated exponentially-distributed data. The Quantile-Quantile plots show that for most of the data, this is an appropriate model. Towards the tail end of the distribution, there is deviation from the exponential, but we believe that these are outliers. Some of the session durations that we observed were in excess of 24 hours. These were probably erroneous measurements, or perhaps players who had mistakenly left their computers connected to a game server, since it is unlikely that someone would be able to play a game for such a long period of time. If the session durations are exponentially distributed, then this agrees with Almeroth and Ammar's

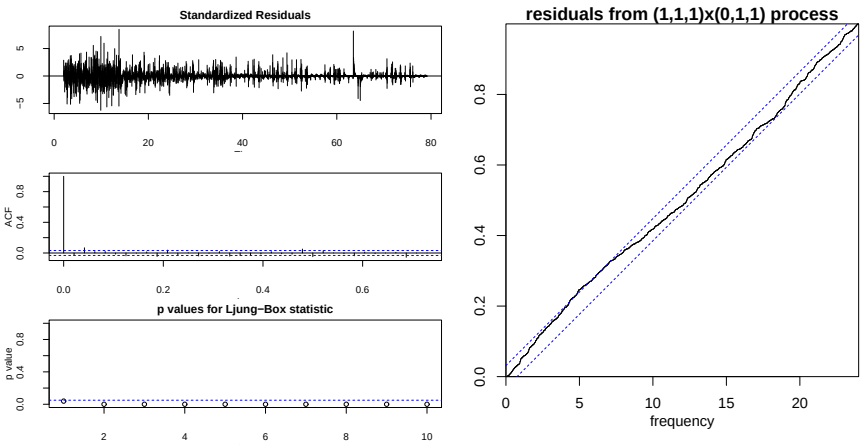


Figure 4.8: ARIMA diagnostics and cumulative periodogram for $(1,1,1) \times (0,1,1)_{48}$ model for a single server

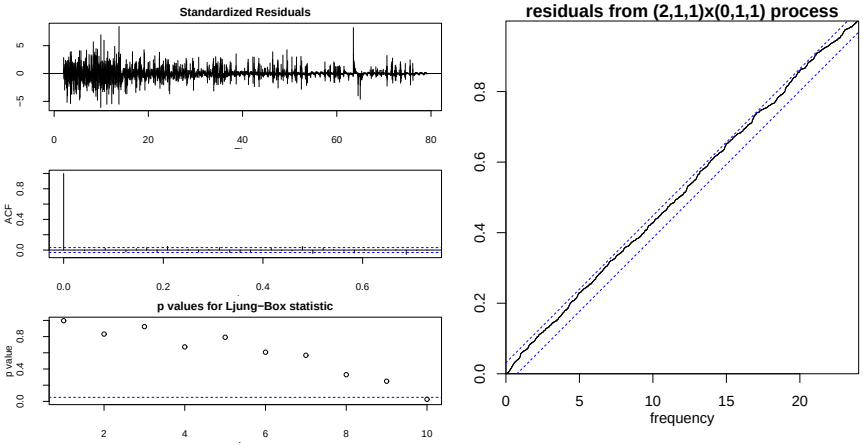


Figure 4.9: ARIMA diagnostics and cumulative periodogram for $(2,1,1) \times (0,1,1)_{48}$ model for same single server as Figure 4.8

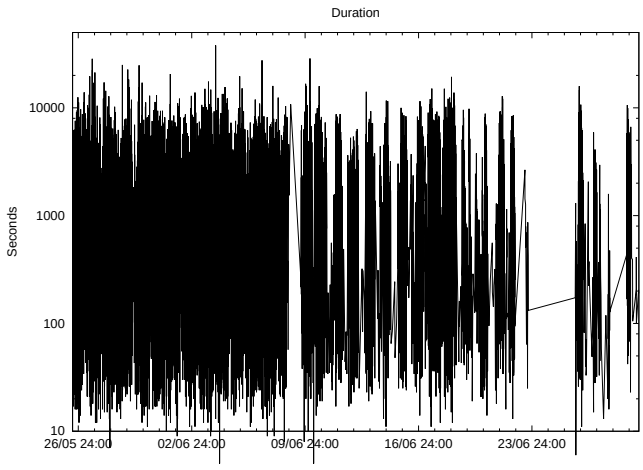


Figure 4.10: Duration of user's game

findings for multicast sessions in [6]. It is also believed that some single-user applications, such as voice telephone calls [30], fit an exponential distribution, although other research indicates that heavy-tailed distributions might be more appropriate [60, 160].

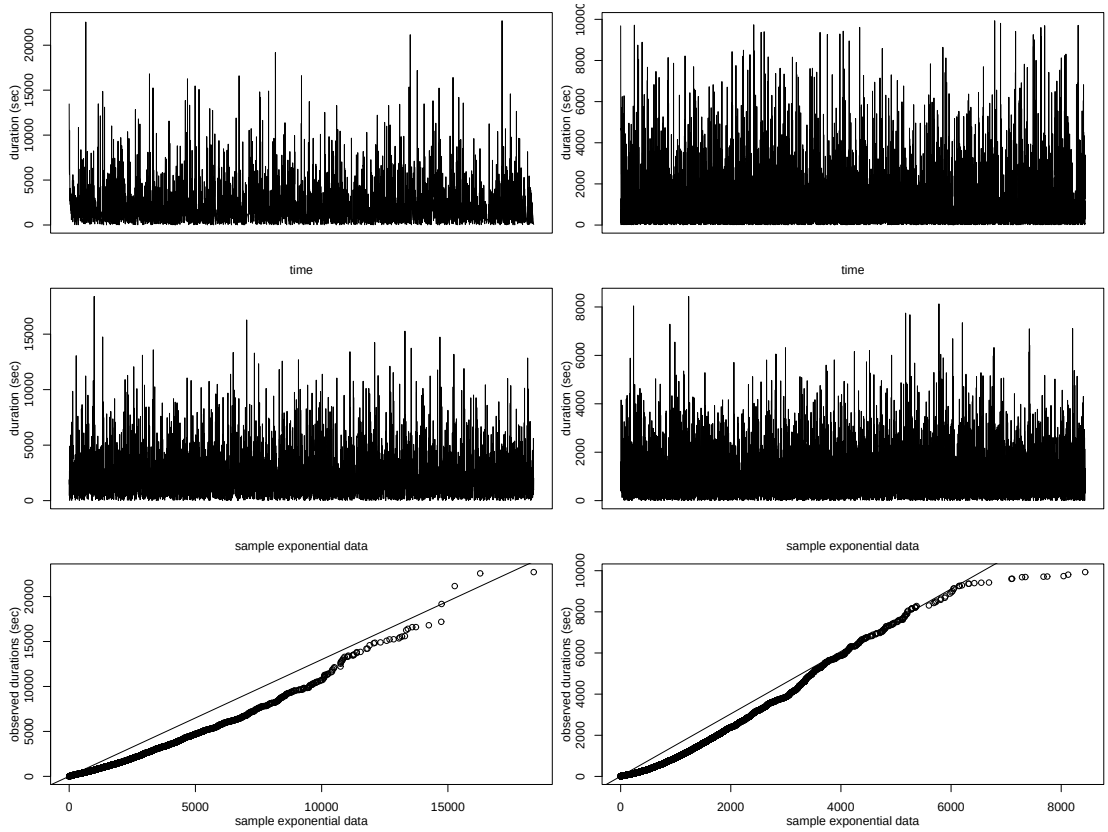
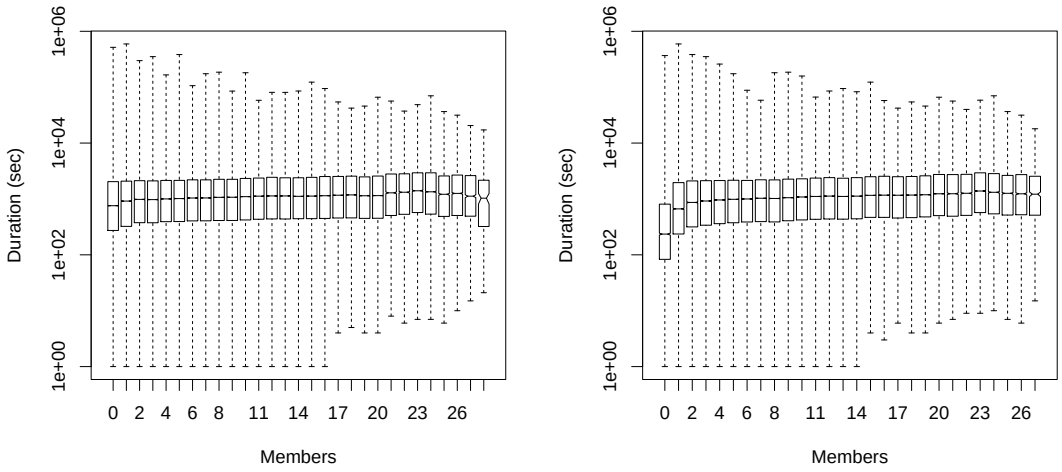


Figure 4.11: Fitting an exponential distribution to user duration data

Since we had already observed network externality effects in the number of players, we expected to find a correlation between the duration of a player's session and the number of players in that game; a game with more players might be likely to lead to players enjoying the game more, which should lead to them staying longer. There appears, however, to be little evidence for this. Figure 4.12(a) shows a boxplot of the number of players at the start of a player's session against the duration of their session. There does not seem to be a very high correlation, and the median duration is relatively constant irrespective of the number of players, $F(1619832) = 2563, p < 0.001, R^2 = 0.002194$. Comparing the duration to the average number of players over the first hour of a session (see Figure 4.12(b)) showed a significant correlation, $F(1619760) = 4170, p < 0.001$, but this correlation was very slight ($R^2 = 0.002568$). This might indicate that the absolute number of players in a session is not necessarily a determinant of when a player decides to leave a session; it may be the behaviour or skill of the specific players that is more important, or a completely unrelated factor.



(a) Number of players at start of session

(b) Average number of players over first hour of session

Figure 4.12: Number of players versus duration of session

4.5 Interarrival times

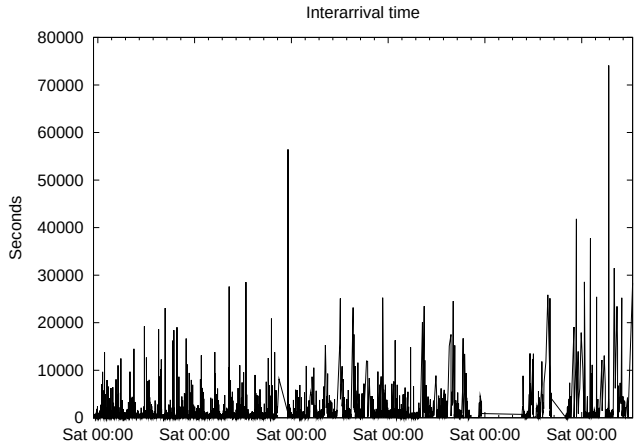


Figure 4.13: Interarrival times

Figure 4.13 shows the interarrival times between players for one server. As for duration, there is large variation. Unlike the duration data, interarrival times do not appear to fit an exponential distribution, as shown in Figure 4.14.

Interarrival times between users for single-user applications have been found to fit a Poisson distribution [67, 160]. This is unlikely to be the case for multiuser applications, however, where the presence of other users may alter user behaviour. Borella [31] and Färber [65] find

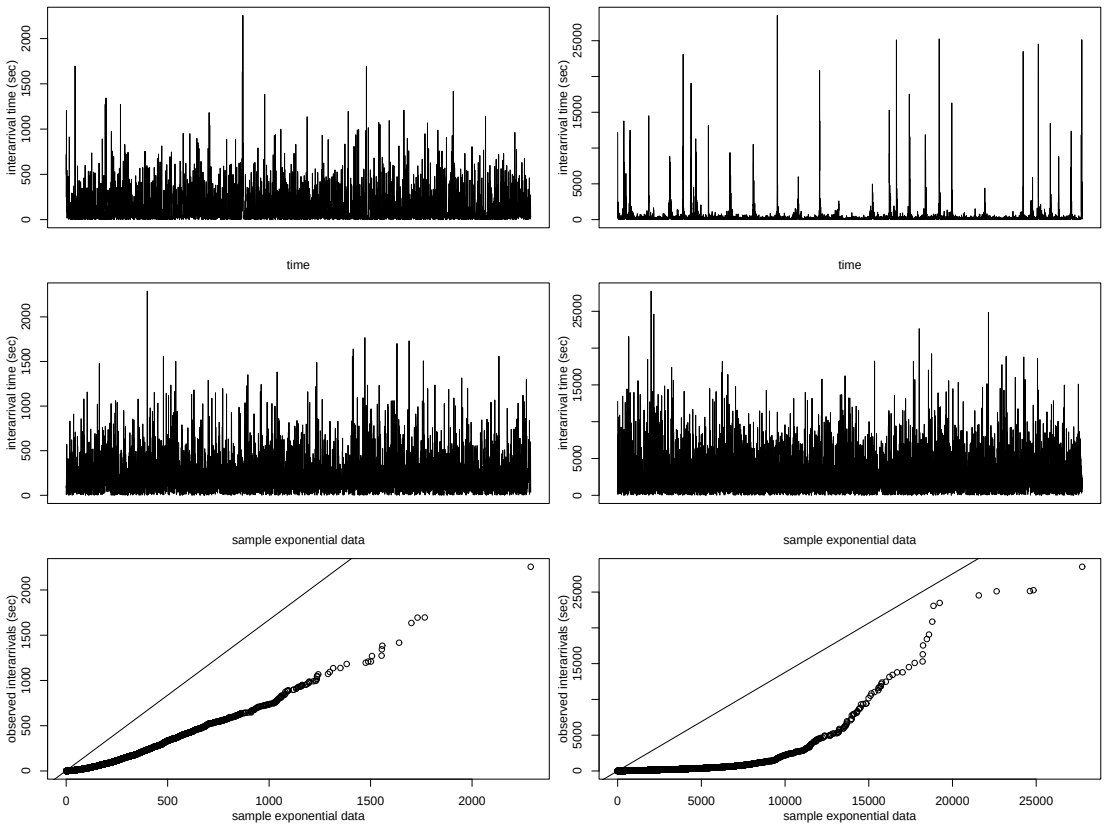


Figure 4.14: Fitting an exponential distribution to interarrival times

that for games, packet interarrival times are highly correlated and bursty. Figure 4.15 shows that this is also true for player interarrivals; there is significant autocorrelation at short lags, which implies that the arrival of some users will lead to others arriving. Thus, the interarrivals do not fit the independent arrivals of the Poisson distribution. This could be the result of network externalities, whereby players observe other players connecting to a game server, and believe that these players must know something good about that server, and thus other players connect to the server, anxious to discover this information. It could also be the result of players joining servers with their friends, for instance, a group of friends may decide out-of-band to play a game on an agreed server, and hence this group would join a server in a short period of time.

Heavy-tailed and Zipf distributions have been observed for Internet usage behaviour, for example in World Wide Web usage [51] and in aggregate Ethernet traffic [204]. One method for visualising a heavy-tailed distribution is a log-log complementary distribution (LLCD) plot, where the complementary cumulative distribution is plotted on logarithmic axes. Linear behaviour in an LLCD plot indicates a heavy-tailed distribution. Figure 4.16(a) shows such a plot for the interarrival times, and linear behaviour can be observed for the larger observations (Figure 4.16(b)).

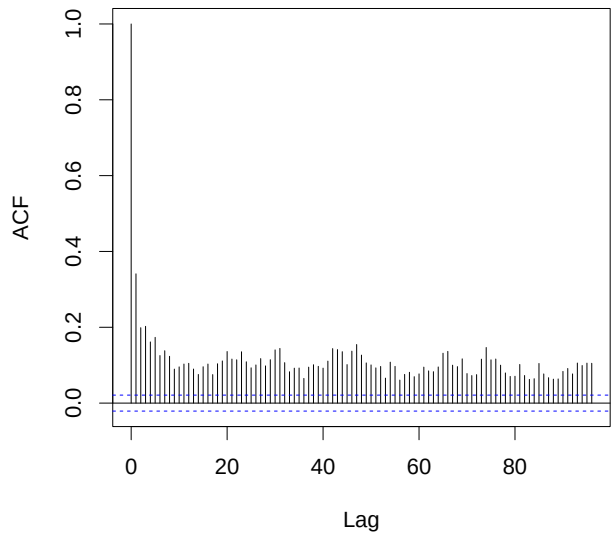
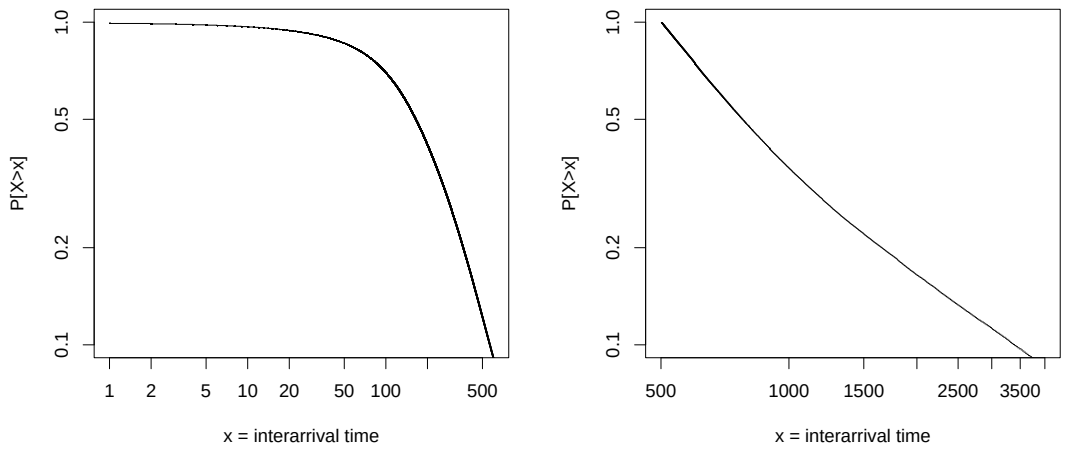


Figure 4.15: Autocorrelation function of interarrival times



(a) full data set

(b) upper tail

Figure 4.16: Log-log complementary plots of interarrival times

A more rigorous test for heavy-tailed distributions is the Hill estimator [91]. A distribution of variable X is heavy-tailed if

$$P[X > x] \sim x^{-\alpha}, \text{as } x \to \infty,\ 0 < \alpha < 2. \tag{4.1}$$

The Hill estimator can be used to calculate α

$$\hat{\alpha}_n = \left(1/k \sum_{i=0}^{i=k-1} (logX_{(n-i)} - logX_{(n-k)}) \right)^{-1} \tag{4.2}$$

where n is the number of the observations, and k indicates how many of the largest observations have been used to calculate $\hat{\alpha}_n$. Figure 4.17 shows that $\hat{\alpha}$ is approximately 1.15.

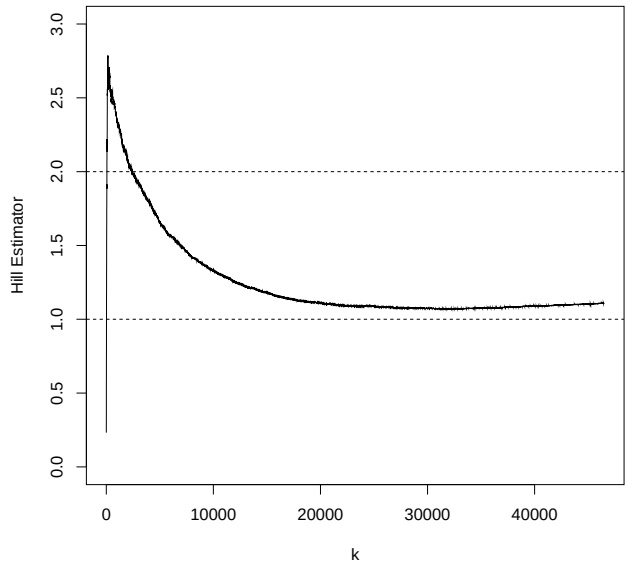


Figure 4.17: Hill estimator for interarrival times

Comparing the interarrival times to the number of players on the server shows some evidence of an inversely proportional relationship (Figure 4.18); as the number of players in a session increases, the interarrival times decrease. This relationship is significant, $F(1605156) = 4.564, p < 0.001$. This supports the hypothesis that the number of players is a determinant in other players' decisions to join a session. Players might use the rate of new joins as an indicator of an exciting or attractive game, as they might assume that the other new players are joining a server as a result of some external knowledge about the server or game. Hence, players might be more likely to join a server which has a high rate of players joining it.

4.6 Summary

In this chapter we have presented statistical analysis of several session-level traces of popular multiplayer networked FPS games. We have found that the number of players exhibits strong time-of-day and network externality effects, and we have fitted an appropriate ARIMA model which incorporates this autocorrelation and seasonal effects. Players' duration times fit an

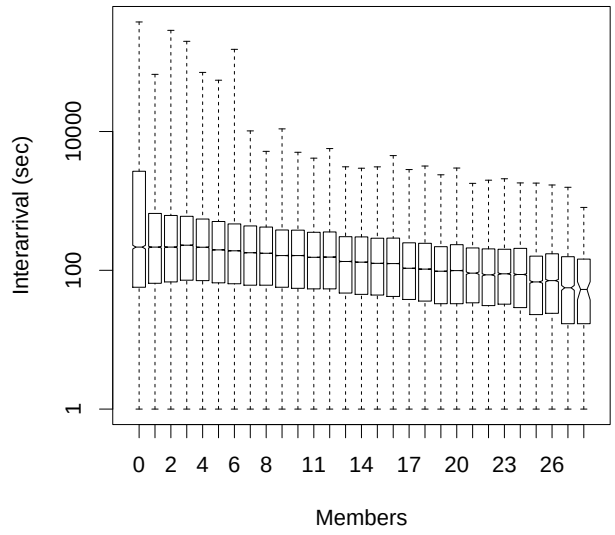


Figure 4.18: Number of players versus interarrival time

exponential distribution, whereas interarrival times fit a heavy-tailed distribution. The number of players in a session appears to have a greater effect on players' decisions to join a session rather than leave. In many respects we have observed similar behaviour to that seen for multicast applications, despite the unicast nature of these games. This implies that in the absence of appropriate multicast data, unicast multipoint applications may be an appropriate substitute.

In this chapter we have demonstrated the following:

- Players consider the number of players in a game when choosing to join a server
- Seasonal and time-of-day effects in the number of players indicate that players prefer to play at weekends
- Session duration may be exponentially distributed
- Player interarrival times are heavy-tailed
- Game session membership is similar to multicast session membership in terms of duration and interarrivals

It appears that the presence of other players on a game server is a factor in a player's decision to join a server. They therefore consider the number of players on a server when deciding to join a game. In the next chapter, we will examine whether players also consider the network conditions of the other players on a game server.

Chapter 5

User behaviour and delay on FPS game servers

5.1 Introduction

In Chapter 4 we demonstrated that game players consider the presence of other players when connecting to a game server. This is indicated by the autocorrelation between the number of players on a server. Thus we can be confident that part of our thesis is true, namely that the group activity that takes place in an FPS game has some bearing on a player's choice of server. This group interaction might also have an effect on a user's tolerances for staying on a server; they may be willing to tolerate lower levels of network QoS in order to continue with an interaction. In this chapter we analyse users' considerations of network conditions when connecting to game servers. In Chapter 4 game players on a large number of servers were examined. Due to the dispersed location and ownership of these servers, it was difficult to gather detailed data about the users on these various game servers. In particular, it was difficult to obtain detailed information about their network conditions. For the work described in this chapter we ran our own game servers, which are publicly-accessible via the Internet. By monitoring the behaviour of players who connect to these servers, and altering the network conditions at the server, more detailed observations and experiments can be carried out. Some of the results presented in this chapter can be found in [85].

The purpose of these observations and experiments was to answer the following questions:

- **Q1** How do game players respond to the existence of network delay?
- **Q2** Do game players consider other users' delay?

If, as the human factors research indicates, network delay is an important factor in the usability of a real-time multimedia application, then we would expect that high network delay would dissuade a user from playing a multiplayer networked game. A player would not choose to connect to a server to which they have high latency, since this would impair their performance

and the playability of the game. In addition, we have already observed in Chapter 4 that the interaction between the players and the number of players has an effect on user behaviour. Thus, we might expect that the network delay of the other players on the server would also have an effect. A player might not want to play on a server where all the other players have a lower delay, since the player with a higher delay might be disadvantaged.

5.2 Methodology

We recorded users that connected to a game server that we set up at UCL in the UK. The game server comprised a 900MHz AMD Athlon PC with 256 Mb of RAM, running Linux kernel version 2.4.2. This was connected to the UCL CS departmental network via 100Base-T Ethernet. The server ran the *Half-Life* daemon version 3.1.07, and was later upgraded to 3.1.0.8 and 3.1.0.9 when these newer versions were released. The software updating was necessary because the corresponding client would only connect to a server which had a equal or greater software version, and so when new versions of the *Half-Life* client were released, the server had to be updated in order to service those players who had chosen to update their clients.

To prevent the possibility of users being prejudiced by connecting to an academic site, we registered a non-geographic .com domain name, `onlinefrag.com`[1], to use instead of a `cs.ucl.ac.uk` address, and the domain name was registered using personal rather than academic contact details. This non-academic address was used for all out-of-band non-gaming communication with users, such as the related website and e-mail contact addresses.

The server was advertised to potential players only by using the game's standard mechanisms, whereby a server registers with a "master server". These master servers exist to provide lists of game servers for players; when a player wishes to play a game, they either connect to a known IP address (obtained through out-of-band information or from previous games), or they query the master server to find a suitable game server. Since no additional advertisement took place, a potential game player should have been unable to distinguish between our game servers and any of the other public servers that are available on the Internet.

The game server was set up to rotate the map, or game level, every 60 minutes, so as to keep the game interesting for existing and potential players. In addition, players were permitted to vote for the next map or to extend the current map at each map rotation interval. The number of players permitted to access the server was arbitrarily set to 24; although the game can potentially support a much higher number of players, most of the more popular maps for *Half-Life* only

[1] A "frag" is the act of killing another player in FPS gaming parlance, and is believed to derive from the "fragmentation" of the avatar that is being killed [98]

effectively scale to 24 players due to a lack of "spawn points" (locations where players can enter the map). There were no specific game-based sessions or goals imposed; players were free to join and leave the server at any time. Thus, when we refer to a **session** in this chapter, we do not refer to a game level or task, or the time between a player entering a level and being killed, but instead a session is the time between a given player joining and leaving a game server.

Player behaviour was monitored at both the application and the network level. For application-level logging, we used the game server API (Application Program Interface) to take advantage of the server daemon's built-in logging facilities (see Section 5.2.2). Packet-level monitoring used the *tcpdump* tool [106], which was set to log UDP packet headers only.

Although *Half-Life* does include features for refusing players admission depending on their delay, and for compensating for variations in player delays [24], these were disabled on our test server, since these might influence any results concerning relative delays and performance.

As the server was only advertised using standard mechanisms, it took some time for potential players to become aware of the server and develop a sufficiently large community of users. Figure 5.1 shows the average weekly number of players on the server over a period of 18 months. It can be seen that it took approximately one month for the server to gain in popularity. There were three periods when the server lost connectivity, due to power outages at UCL. These are the three drops in membership in Figure 5.1. After each of these outages, it took between one and three weeks for session membership to return to its previous levels. For the purposes of this study, we only examine the periods when session membership was in a steady state.

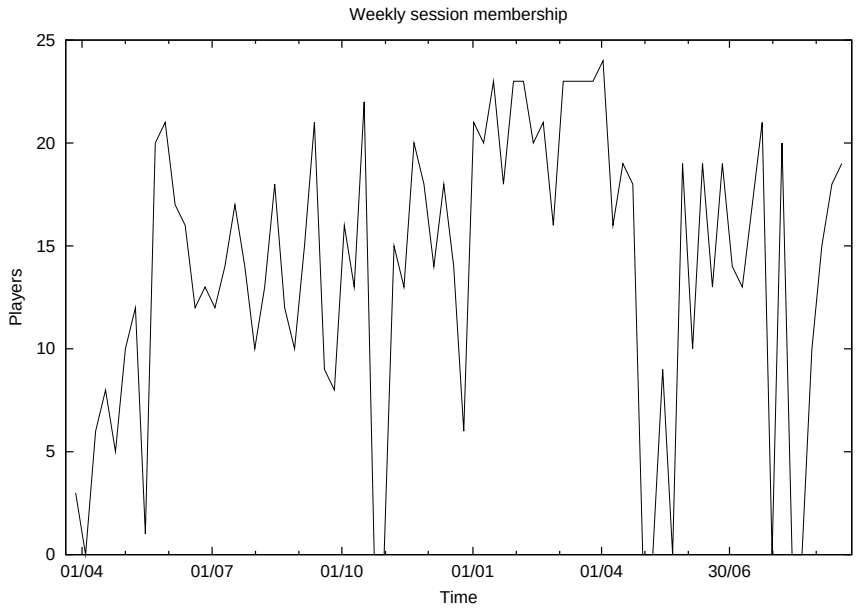


Figure 5.1: Weekly session membership on a *Half-Life* server

5.2.1 Determining unique users

Many of the issues that we wished to examine require knowledge of which sessions correspond to which particular users, for example, to allow us to determine the average delay observed by a particular player across all of their sessions. Such persistent user/session relationships cannot be determined by network-level traces alone, and session-level data is required. The nature of most FPS games, however, where any user can connect to any available and appropriate server with a minimal amount of authentication, means that determining which sessions belong to which users can be difficult.

Connecting to a *Half-Life* server is a two-stage process. The client first authenticates with the so-called "WON Auth Server" (the acronym WON stands for World Oppponent Network, the organisation that runs the gaming website `http://www.won.net`, which is owned by Sierra Software, the publisher of *Half-Life*). The authentication server issues the player with a "WONID", a unique identifier generated using the player's license key, which is provided with the CD-ROM media when a user purchases the *Half-Life* software. There is thus one unique WONID for each purchased copy of the game, which should correlate to one unique WONID per user. Once a WONID has been generated, the player can connect to the *Half-Life* server of their choice.

Unfortunately, using the WONIDs as a means of identifying unique players proved to be insufficient. We observed a large number of duplicate WONIDs, indicated by simultaneous use of the same WONID, or players with the same WONID connecting from highly geographically dispersed locations. This duplication of WONIDs is probably due to the sharing of license keys, the use of pirate copies of the game, or malicious worms or viruses which have been specifically designed to steal these keys from unsuspecting users [189]. A single WONID can thus represent more than one user. This situation is exacerbated because the game server program does not reject multiple users with the same WONID from playing simultaneously (this occurred 1,924 times during the period of this study). In addition, on two occasions the WON Authentication Server appeared to malfunction, and issued all users with a WONID of 0. Although it would have been possible to modify the server to reject simultaneous duplicate WONIDs, this would not resolve the problem of different players connecting at different times with the same WONID, and so rather than reject players, a scheme was needed to determine which sessions belonged to which different individuals.

For each player that connects to the game server, the following information is logged by the game: the player's WONID, their IP address and port number, and the nickname that they have chosen to use in the game. Of the 49,667 total WONIDs that were observed, 36,820 had unique

(WONID, nickname, IP address, port) tuples, and we can be reasonably sure that each of these represents one unique user. In order to determine their uniqueness of the remaining WONIDs, we had to make some assumptions about users. We assume that a unique (WONID,nickname) tuple across all sessions is a single user, and probably has a dynamically-assigned IP address or multiple access ISPs. In Chapter 4 we used a player's nickname to distinguish unique users. By analysing the server logs, we found that users might occasionally change their nicknames, both between and during sessions, and by looking at all the names used by a particular WONID and looking for common names it was possible to further isolate potential unique users. For instance, a user might connect with the default name set by the game ("Player"), and then realise this and later change their name to their usual nickname, or a player might change their name to signal other users, e.g. new players would sometimes change their name to "Please don't shoot me" or the like. When multiple users with the same WONID were simultaneously connected we assume that each of these is a different user. The country of origin of each player was estimated using whois records (see Section 5.3), and when users with the same WONID connected from different countries on the same day, these were also assumed to be different users.

5.2.2 Measuring delay

To measure the delay observed by each player we used the game server's built-in facilities. As previously discussed, a user in a typical FPS game has the ability to view the "scoreboard", which indicates how well the player is doing in relation to the other players, by a metric such as the number of times that they have killed the other players. In *Half-Life*, this scoreboard also includes a "ping" value, which is an indication of the player's round-trip time to the server. This measurement is taken at the application-level, calculated between the client and server applications.

The game server's logging facilities were modified to record the application-level delay of all the players every 30 seconds, and to take an additional measurement whenever a player joined or left the server. Of the total 6,805,217 measurements, we removed the 69,224 measurements with a ping value of 0, assuming that they were errors. We also saw 50,667 measurements greater than 1,000ms, with a maximum of 184,693ms. We removed these measurements, also assuming that they were errors, since it is unlikely that any user would be able to play a networked game effectively with a delay of over one second, and certainly not over three minutes. Moreover, a similar FPS game, *Quake III Arena*, also assumes that client delays over 1,000ms are errors, and chooses not to report them to users at all.

We did not measure network-level delay, e.g. through ICMP pings, since one of our experimental design criteria was that we did not want to alter the server in any way, or send additional traffic to players, in case this altered player behaviour or deterred some potential players from connecting to the server. With over 15,000 other potential servers to choose from, we did not wish to alter conditions in such a way that players might be driven elsewhere. We initially attempted to use an active measurement system, taking network-level measurements through the use of ICMP ECHO_REQUEST ("ping") packets. When these network-level measurements were attempted, users either detected these and complained, or filtered ICMP traffic — it is common for systems adminstrators to filter ICMP traffic for security reasons, for instance to prevent "smurf" denial of service attacks [38]. We therefore chose to use a passive measurement system, by using only the measurements that are already provided by the standard game client and server software. In this way, no additional traffic was required to be generated, and so users would not notice this and alter their behaviour. This has some benefits in that it is the application-level delay which the users themselves observe, and thus one would expect that this would have a greater effect on their behaviour than network-level delay.

Since measurements were being taken at the application level, we would expect that these measurements would be higher than those taken at the network-level. To estimate the accuracy of this application-level delay measure, we obtained a list of *Half-Life* master servers. These were queried every six hours to produce a list of available *Half-Life* game servers. Each of these servers was then queried using the game-specific server query protocol — we used a game-specific command to query the application-level delay ten times. In addition to this, ten standard ICMP ECHO_REQUEST packets were sent to query the network-level delay of each server that permitted ICMP traffic (10.75% of the servers did not permit ICMP traffic). Every six hours, both of these queries were performed, for a total of 161,755 measurements taken over the course of one month. By looking at the measurements for the servers which allowed ICMP traffic, a comparison of the application-level and network-level delay measures can be made. Figure 5.2 shows that the correlation between the two measures is very high — the correlation coefficient is 0.925. As expected, the application-level delay is typically higher than the network-level delay measure (this is not always the case since the two measurements are not taken at exactly the same time, and also because some Internet routers might place ICMP traffic on a different path to UDP and TCP traffic). On average, the application-level measure was 7.24% higher than the network-level measure. We can therefore be confident that the application-level measure provides an appropriate indication of the latency observed by a game player.

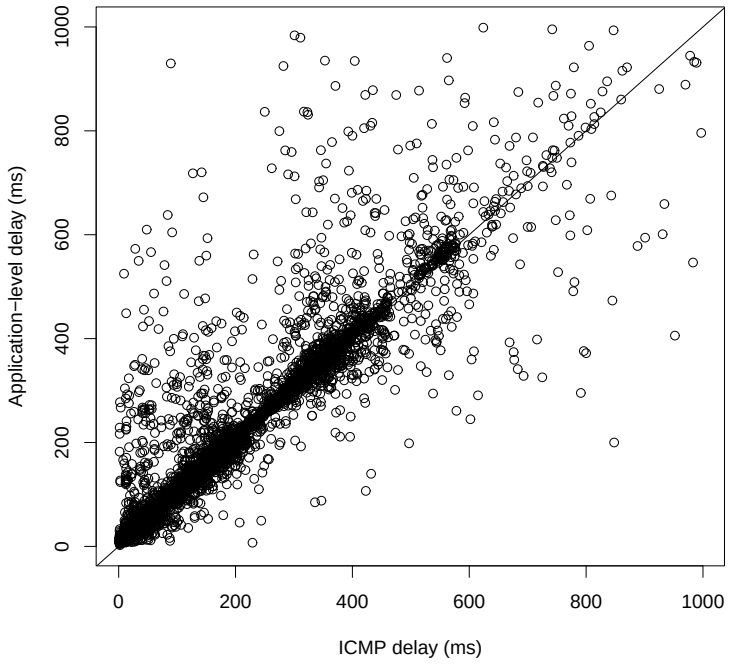


Figure 5.2: Correlation between application-level and network-level delay measurements

5.2.3 Measuring relative delay

Absolute delay bounds might not be that important because players become accustomed to high delays, or they have no choice because they happen to have poor network connectivity. A more important delay metric might be the relative delay between players. If one player has a much lower delay than the other players in the game, they might be able to exploit this advantage, by attacking players before they are able to respond.

Relative delay may be perceived by a player in a number of ways. A player might dislike it if the other players on a server have a wide range of delays — for instance there might be some players with very low delays, and others with extremely high delays, which might make a player with a intermediate level of delay unsure about the responsiveness of the other players, since they would not know whether to expect a player to respond very quickly, or very slowly. A player might also be worried about relative delay if they have a very different delay to the other players on the server. This could take the form of having a much higher delay than the player with the lowest delay on the server, or by having a higher delay than the average player on the server.

With this in mind, we chose four metrics with which to measure the relative delay between the players. We refer to these as follows:

1. *stddev* — the standard deviation of the delay experienced by all the players in a given user's session.

2. *avgratio* — the ratio of a user's delay compared with the average delay of all the players on the server.

3. *topratio* — the ratio of a user's delay compared with the player with the lowest delay on the server.

4. *rank* — the "delay rank" of the user compared with the other players on the server.

The first three measures of relative delay, *stddev*, *avgratio* and *topratio* are nominal:

- for a given player *i*, *stddev* is calculated by taking the average delay observed for each of the players who connects to the server during player *i*'s session. *stddev* is the standard deviation of all these delay values.

- *avgratio* is *i*'s average delay, divided by the average delay of all the players who connect to the server during *i*'s session.

- *topratio* is *i*'s average delay, divided by the average delay of the player with the lowest delay of the players who connect to the server during *i*'s session.

rank is an ordinal measure. *rank* was calculated by ordering the players at each delay measurement period by delay to produce a rank *r*, and then normalising this according to the number of players on the server. Thus, the player with the highest delay would always receive a delay rank of 1, whereas the player with the lowest delay would have a rank of 0. The intuition for examining an ordinal measure of relative delay was that a player might not care about the exact ratio of their delay compared with that of the other players, but they might care that their delay is higher than a certain proportion of the players on the server. For instance, a user might be able to play on a server when they have a delay twice as high as that of the lowest player on the server, so long as they have a lower delay than more than half of the remaining players. Additionally, the question of whether users' preferences and utility functions are cardinal or ordinal is a long-standing debate in economics and one of the foundations of neoclassical economic theory, and it is unclear whether absolute or relative values should be used to calculate user preferences.

5.2.4 Inferring user preferences

User preferences can be measured implicitly or explicitly. McCarthy [137] outlines the difference between these two types of measurement: "there is a tradeoff between implicitly infer-

ring preferences, which may be inaccurate and perceived as invasive, and explicit requests for preference information, which may be burdensome and imposing (and therefore not used by some/most inhabitants)". Since the measurements were taken on a passive basis, we did not directly ask users whether they enjoyed a game or not. Thus, we chose to implicitly infer users' preferences for a game. There are three occurrences which can be analysed, from which we can infer user preferences:

- Whether a user joins a game server
- How long a user plays on a game server
- When a user leaves a game server

We can infer that a user enjoys a game if:

- A user joins and remains on a server
- A user repeatedly returns to a server

Conversely, we can infer that a user dislikes a game if:

- A user joins and immediately leaves a server
- A user leaves a server because of a change in conditions

Therefore, we can infer the preferences of the users by:

- comparing the players who join and immediately leave a server to those who join and remain on the server
- examining the network conditions which lead a user to stay on a server
- examining the network conditions when a user leaves a server

We measure how long users remain on the server, and how often users return to the server. We use this information to define two classes of user:

- **"regulars"** — players who return to the server more than ten times, and whose average session duration exceeds one minute
- **"tourists"** — players who connect to the server for less than one minute

5.3 The user population

The data that is analysed here derives from running the server between 21 March 2001 18:33 GMT and 13 September 2001 11:06 BST. In this time we observed 133578 sessions (a single user joining and leaving the server). Using the heuristics described in Section 5.2.1 to determine unique users, we estimate that the 49,667 unique WONIDs that were observed actually represent 79,880 different users, due to cheating and the sharing of license keys.

A simple method for determining the general geographical location of users was used. First, a DNS query on each player's IP address was performed. For those IP addresses that resolved to hostnames with geographical Top Level Domains (TLDs), we assumed that this TLD is the player's country of origin. For the remaining IP addresses which either failed to resolve to a hostname, or resolved to non-geographic TLDs (.com, .net, .org, .mil, .edu, .gov and .int) a whois [84] query was performed, and we assume that the country listed in the whois database is the player's country of origin. To save time, DNS and whois queries were cached, and to reduce load on the whois servers we used the Regional Internet Registries' list of address allocations [171] to get an idea of which whois server to query first (although there are techniques for whois servers to refer queries to the appropriate server [203], these do not seem to have been implemented by many whois server operators). Problems with using TLDs, which do not necessarily reflect the geographical location of a host, are well-known [146]. We have tried to account for some of this noise by assuming that some of the geographic TLDs which are available for registration by residents of any country, such as .nu, .to and .tv, are non-geographic, and used the whois database for these. On the other hand, relying solely on the whois databases can also result in erroneous conclusions. Many multinational ISPs have the same whois entries for all their national subsidiaries, and a reverse DNS lookup on the IP address can help to distinguish between, for example, Libertysurf's British and French users (Libertysurf provides all of its users with addresses that resolve to hostnames ending in `libertysurf.fr`, but for some of these addresses, the whois database entries locate them in the UK). Moreover, more comprehensive methods for determining the geographical location of a host, such as GeoPing [154], require intrusive measurements such as query packets directed at the hosts in question, and would have been inappropriate for our passive measurement system.

There were a total of 39,955 IP addresses observed on the server. Figure 5.3 shows the total for each inferred country of origin for these addresses, and for the unique players observed on the server. Although the server was located at UCL in the UK, both the majority of the IP addresses that connected to the server, and the actual players, appear to come from the USA. This predominance of US players may be due to JANET's (the UK's academic network) large transat-

lantic links (6x155Mb/s links from Teleglobe in London to Teleglobe in New York [193]). Only 1.12% of the US addresses, however, resolved to academic `.edu` hostnames, so JANET's connectivity to the American academic networks does not seem to have been an influencing factor in player behaviour (although we lack statistics for the overall incidence of gameplay amongst academic and non-academic networks). This is also true for the UK, and of the 12,514 UK addresses we saw, only 27 of these appeared to be from academic institutions (i.e., had hostnames which ended in `ac.uk`). This might perhaps be due to successfully-enforced university regulations against playing games. Whilst perhaps surprising, since we might expect more academic users on servers which are connected to an academic network, this low proportion of academic users means that our population sample can be considered more representative of Internet users as a whole.

After the USA, the UK has the second-highest number of players on the server. This is what we would expect, since the servers are located in the UK. Surprisingly, the fourth-highest number of players comes from Taiwan, despite Taipei being physically located over 6,000 miles from London and players from Taiwan having an average delay of 475.12ms.

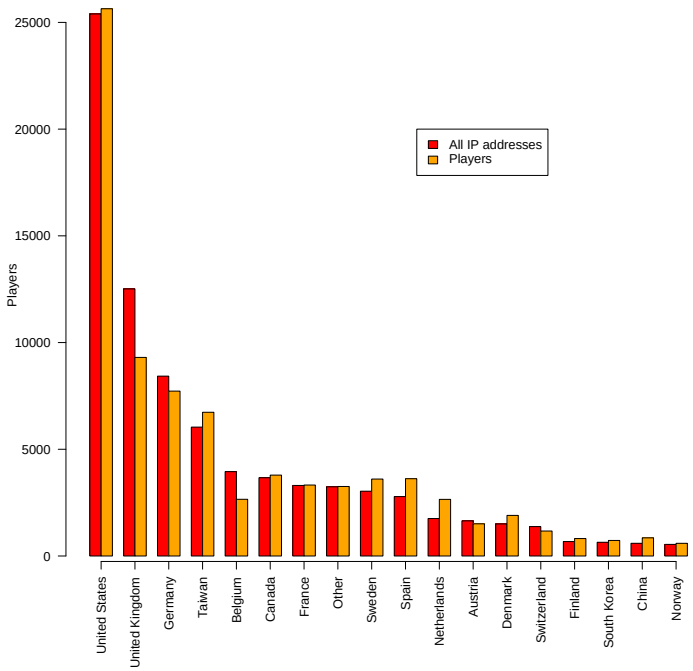


Figure 5.3: Location of all observed IP addresses

Removing "tourists" from the data shows that US players still dominate. Figure 5.4 shows the location of players whose average session duration was less than one minute, and those

who played more than ten times. It can be seen that most Taiwanese players stay for less than one minute — it would appear that they are happy to browse servers, but then they leave, perhaps due to their higher propagation delay compared with the geographically-closer US and European players. This behaviour is strange, however, given that we expect that most players tend to select a server through a server browser (although there is such a browser built into the *Half-Life* client, which is shown in Figure 2.4, players may also choose to use third-party utilities such as *GameSpy* [75] or *The All-Seeing Eye* [4] which can query a number of different games simultaneously). Such server browsers estimate the delay to a given server before the player actively connects to it. The fact that players connect to servers in spite of high delays might indicate that the composition of the game (i.e., the current players and their behaviour) is more important than an individual player's delay.

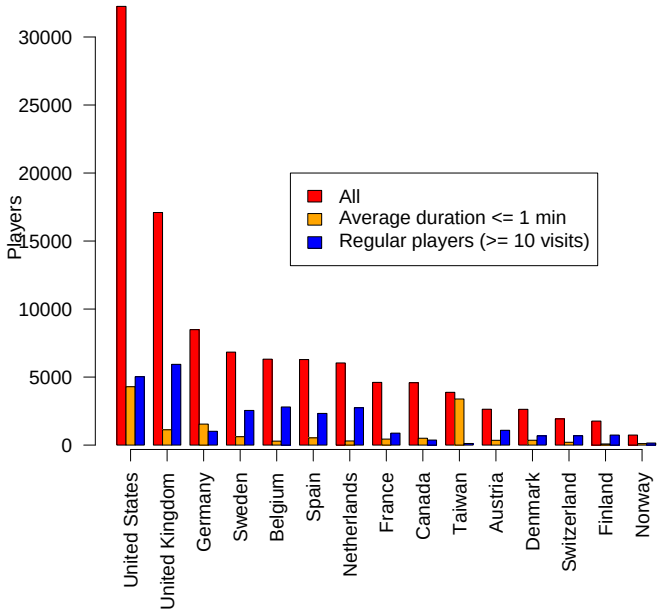
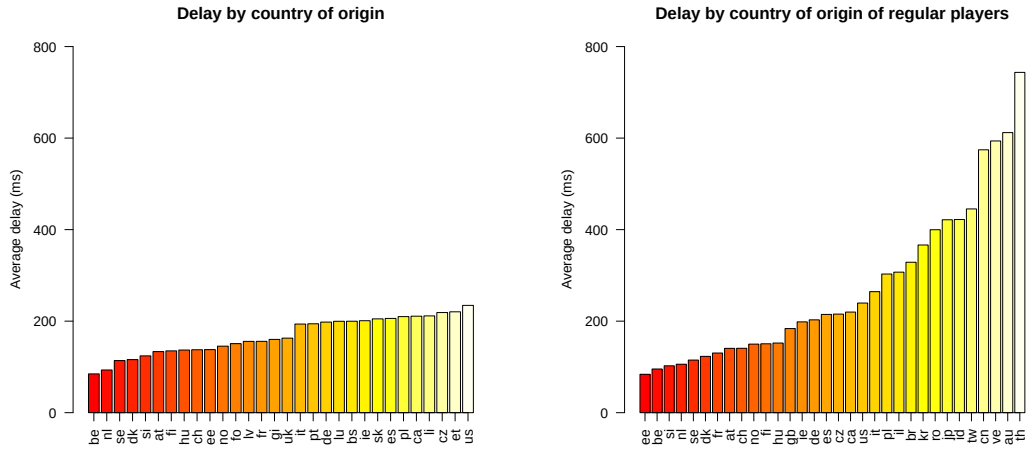


Figure 5.4: Location of "tourist" and "regular" players

Examining the location of only the regular players who played more than ten times (Figure 5.4) shows a larger proportion of European players, as we would expect. There is still a large number of players from the US, however, in spite of the added transatlantic network delay, although the US does not make up the largest single country of origin for regular players. The Taiwanese players are again interesting — although many Taiwanese players connect to

the server, it appears that most of these players then leave and do not remain for more than one minute.

The delays of players from different countries also fail to explain the prevalence of American players. Figure 5.5(a) produces few surprises; European players have lower delays than those from North America. Players from the UK, however, appear to have higher delays than those from most other European countries, in spite of their geographical proximity. The same is true even when looking at the regular players (Figure 5.5(b)). The high delay for the regular players from .th (Thailand) is attributable almost entirely to one player, who in spite of a minimum delay of 676.87ms, was seen connecting to the server 18 times with an average duration of 4022.88 seconds. Strangely, the UK is not the country with regular players with the lowest delay. The country with the lowest delay is Estonia, and average players from the UK have a relatively high level of delay of 183.79ms.



(a) Delays of all players by country (limited to 250ms)

(b) Delays of regular players

Figure 5.5: Delay of players sorted by their country of origin

One reason for the large number of US players might be that this is a reflection of the high proportion of US-based hosts on the Internet. To determine whether this is true, we compare the proportion of addresses from each TLD with the proportions seen in the Internet Domain Survey (IDS) [103] (we ignore TLDs observed less than 3 times to avoid a bias against non-significant results). Figure 5.6 shows that in fact we see a smaller than expected proportion of addresses from non-geographic TLDs such as .net and .com, and a higher proportion of addresses from

European TLDs such as .uk and .de. There are, however, also a higher proportion of addresses from several non-European TLDs, including Japan and Canada. It is therefore unclear whether the high number of US-based players is due to the predominance of US hosts on the Internet. Moreover, it should be noted that since the IDS only estimates the location of hosts by hostname, it might not be as accurate as the DNS/whois system that we have used. For instance, for the estimation of hosts from Taiwan, only 19.03% of the Taiwanese addresses actually resolved to a hostname ending in .tw, with 38.76% of the addresses not resolving at all, and the remainder resolving to non-geographical TLDs.

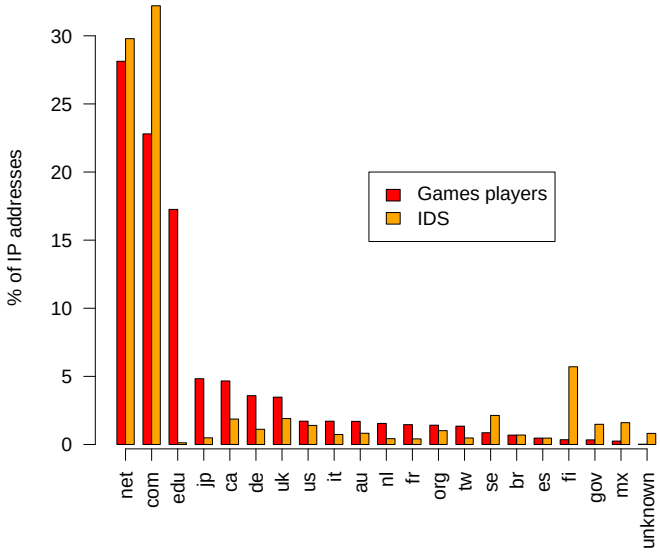


Figure 5.6: Observed TLDs compared with the IDS

Another reason for the number of US players might be a lack of available game servers in the USA. To determine whether this might be the case, the master *Half-Life* servers were polled from two sites in Europe and the USA every six hours for a period of nine months. Each game server on the list provided by the master servers was then polled to determine the application-level delay between the server and the polling site. The servers that could be reached were ordered by delay, and then the ranking, i.e., the position in this ordered list, of the two *Half-Life* servers at UCL were calculated. This methodology should be representative of a player using an in-game server browser to choose a server.

Polling site	**Europe**		**USA**	
	Mean	Median	Mean	Median
Number of servers on master server list	22574.09	22586	22516.71	22529
Number of reachable game servers	6430.22	6521	6534.85	6645
Delay to `hlife.onlinefrag.com`	14.72	8.70	110.09	88.69
Rank of `hlife.onlinefrag.com`	253.40	62.00	2825.97	2513.50
Delay to `hlife2.onlinefrag.com`	14.19	6.60	112.04	92.89
Rank of `hlife2.onlinefrag.com`	234.09	64.00	2780.51	2530.00

Table 5.1: Available servers in Europe and the USA

Table 5.1 indicates that, as might be expected, there were approximately 2500 servers with lower delay than the servers at UCL available to the US site. The polling site in the USA was based on the east coast (`zealand.nge.isi.edu`, located in Washington, DC), and we expect that the number of servers with lower delay would be higher for sites elsewhere in the USA. The number of game servers reachable by the two sites was similar, at just under 30% of the total number of servers advertised by the master server (this figure is remarkably low, and possible reasons for this are discussed with further details in [86]).

To summarise, the user population that we have observed on the game servers predominantly come from America and Europe, in spite of the propagation delay incurred by the transatlantic crossing. The high number of American players cannot be explained by a lack of game servers in the USA, nor by the predominance of US-based hosts on the Internet. One possible conclusion is that the drawbacks of the high delay are compensated for by other factors, such as the interaction between the players on the server.

5.4 Joining a server

In this section we analyse the conditions which lead a user to decide to play on the game server. In particular, we analyse user behaviour in the first minute of a session. If a user leaves within one minute (i.e., a “tourist”), we infer that they did not find the server appropriate. This one minute boundary was chosen arbitrarily, but we believe that one minute should be long enough for a user to decide whether or not they intend to stay on the server.

Figure 5.7 shows the delays of all the players that were observed on the server, including the delays of “tourists” — those players who connect to a server, examine the status of the game and then choose to leave. Figure 5.8 shows the distribution of delay for all the players compared with those who stay less than a minute, those who stay more than ten minutes, and those who

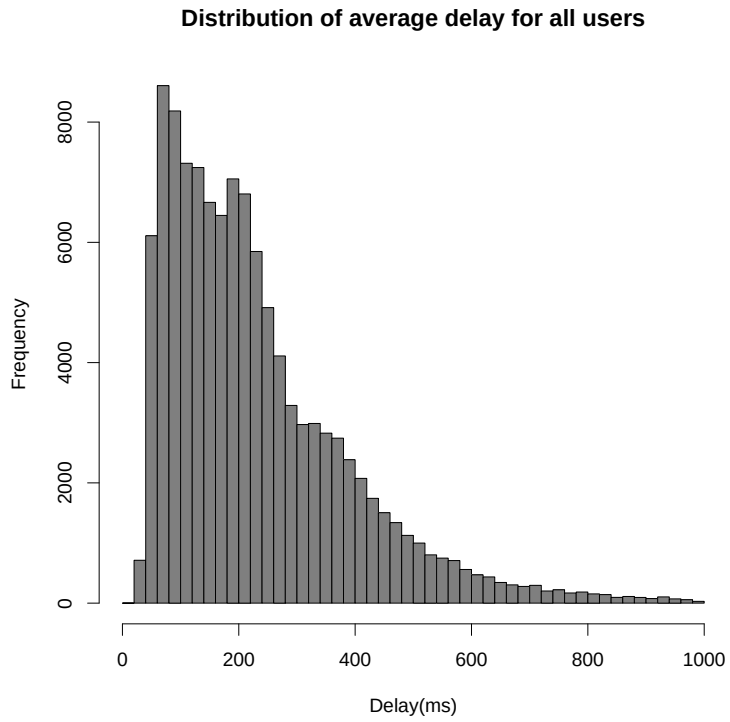


Figure 5.7: Distribution of players' average delay

Players	**Mean delay (ms)**	**95th percentile**	**25th percentile**	**50th percentile**	**75th percentile**
All	231.70	546	112.25	194.29	308.0
Regular	144.02	342.45	69.35	113.89	197.19
Tourists	294.45	664	149.38	244	410.92

Table 5.2: Overall delay results

play for in excess of one hour. It can be seen that the delay of those players who stay less than a minute is generally higher than those who stay for longer. Of the players who connect with a delay of over 400ms, only 10.09% stay for longer than one minute. This difference in delay between the tourists and other players is significant, $t(6462) = 26.91, p < 0.01$. This implies that delay may be a determinant in a player's decision to join a server; players with high delays to a particular server will join, observe their high level of latency, and then choose to leave the server and perhaps look elsewhere for a more appropriate game server.

To further test whether users can detect and respond to the presence of network delay, two identical *Half-Life* game servers were set up, comprising 1.2GHz AMD Athlon PCs with 256 Mb of RAM, running Linux kernel version 2.4.9 and *Half-Life* version 3.1.0.8. These were

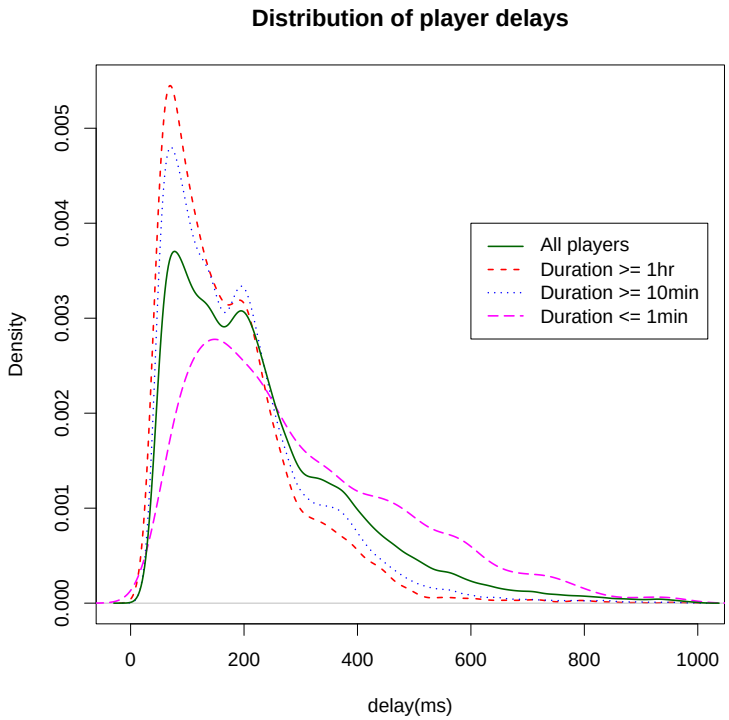


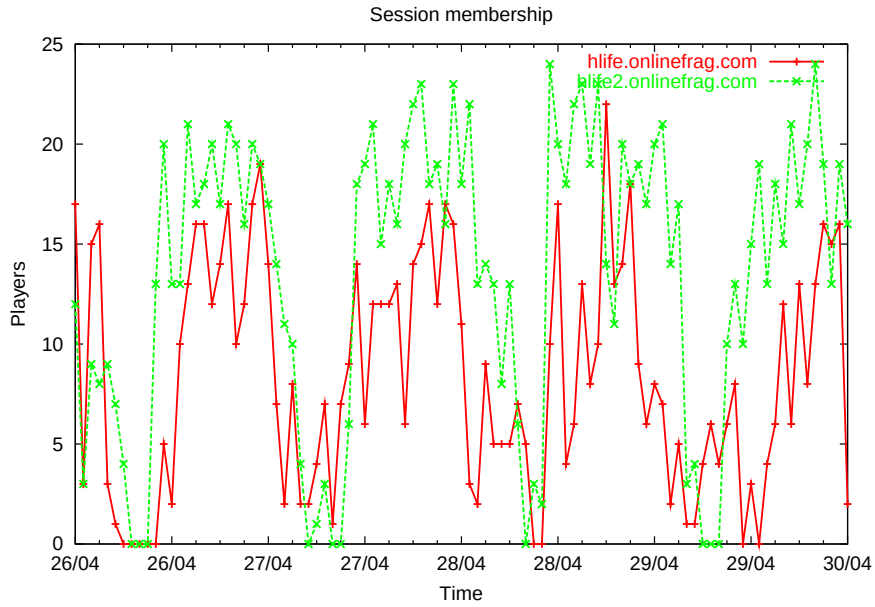
Figure 5.8: Kernel density function of players' delay

connected to the public Internet via a gateway machine, which was used to introduce delay into the network. The gateway machine was a 1GHz AMD Athlon PC, also with 256 Mb of RAM and running Linux kernel version 2.4.9.

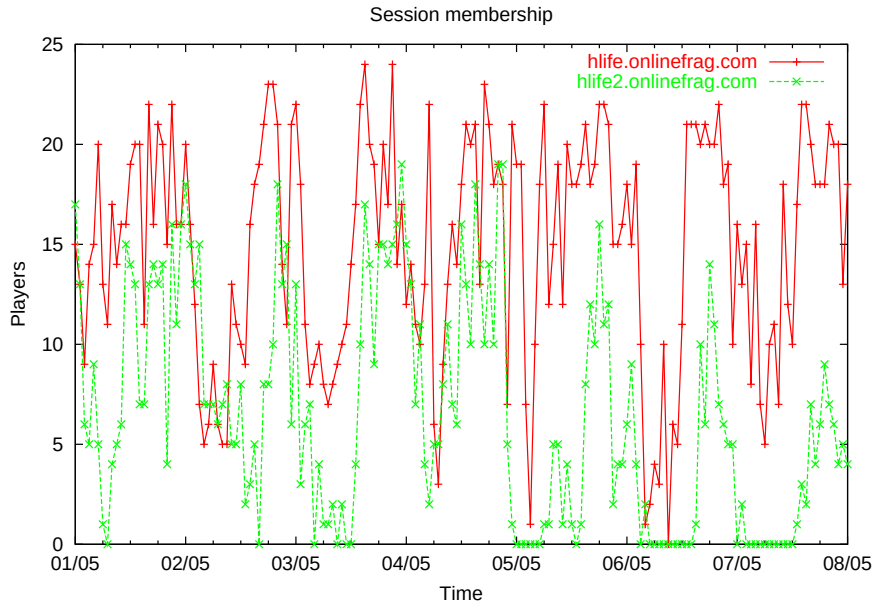
The *iptables* and *libipq* interfaces in Linux [148] were used to introduce delay into the network. An *iptables* filter was set up to queue packets which were addressed to the IP address and port number of the game server. These packets were queued in userspace and placed back on the queue after the desired period of time had passed. Initial experiments attempted to use the *Dummynet* [172] and *NISTNet* [151] network emulation packages for FreeBSD and Linux respectively, but these were found to be inappropriate. *Dummynet* lacked sufficient time resolution, and could not provide an accurately controllable amount of delay, whilst *NISTNet* was unreliable, and as it ran in kernel-space, would lead to the gateway machine completely freezing in the event of a crash. This was undesirable since experiments would run for weeks at a time, and if a machine crashed and was not rebooted immediately, the lack of available servers would deter players, as discussed in Section 5.2.

The servers were left to run for two months to build up a regular userbase. In the month following this, 50ms of additional delay was introduced on to one of the servers. This additional

delay alternated between the two servers, so that users would not begin to ignore one of the servers.



(a) Additional delay to hlife.onlinefrag.com



(b) Additional delay to hlife2.onlinefrag.com

Figure 5.9: Players on two servers with differing levels of network delay

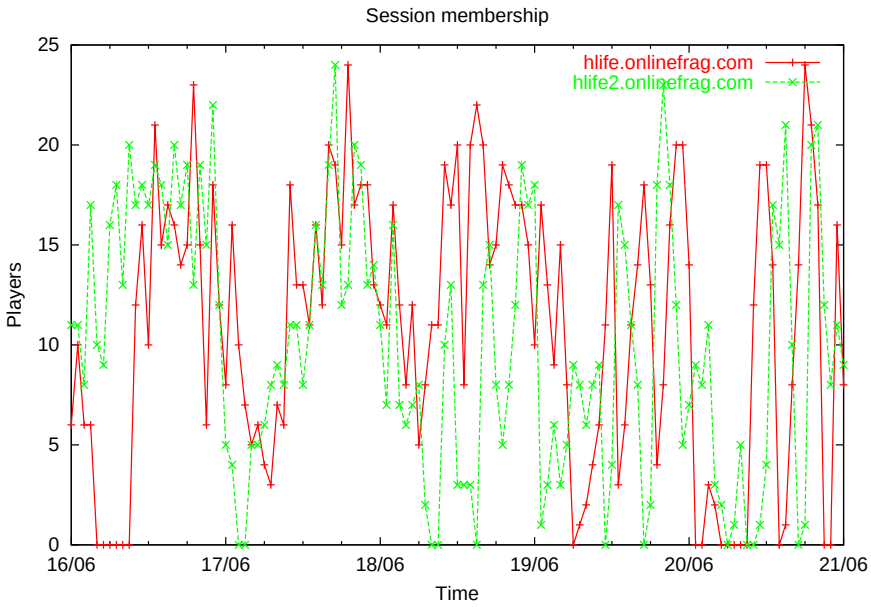


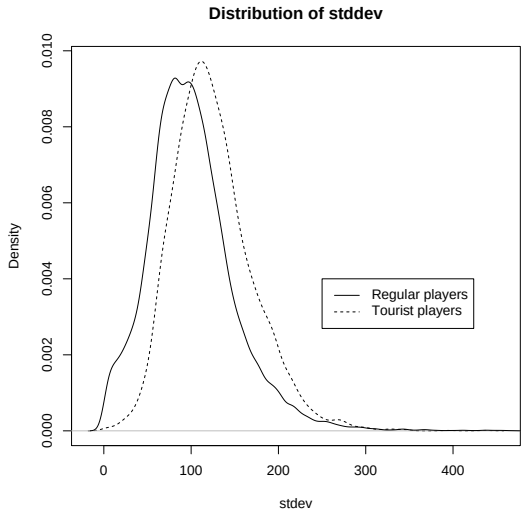
Figure 5.10: Players on two servers with no additional network delay

Figures 5.9 and 5.10 show the number of players on the two servers during a representative sample of the experiment (the data in the two graphs have been windowed by a 60 minute period for clarity). Figure 5.9 shows that in the presence of additional delay, the number of players that connect to a server drops markedly. From 26/04/02 to 02/05/02, `hlife.onlinefrag.com` (the solid line) had additional delay, and so there are fewer players on that server (Figure 5.9(a)). From 02/05/02 to 08/05/02, `hlife2.onlinefrag.com` (the dashed line) had additional delay, and the situation reverses (Figure 5.9(b)). The data were windowed to create a time-series with a frequency of one minute. A paired t-test on this time series indicates that the difference between the number of players on the two servers is significant, $t(2016) = 59.65, p < 0.01$. Figure 5.10 shows that by comparison, in the absence of any additional delay, the difference in the number of players on the two servers is insignificant, $t(4591) = 1.66, p > 0.05$. We can thus conclude that network delay does have an effect on a user's decision to join a game server.

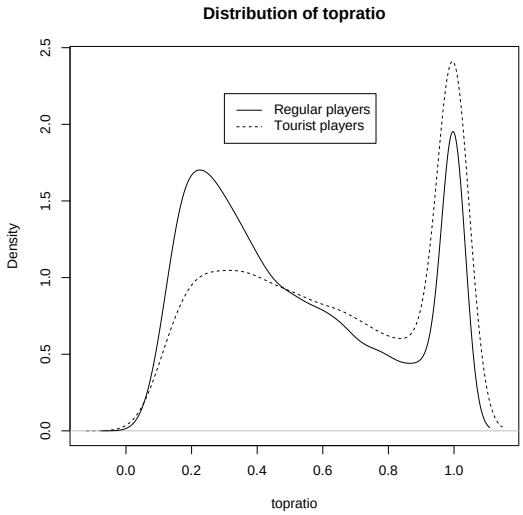
5.4.1 Relative delay

As with absolute delay, we compare the relative delay metrics for the tourist and regular players, to determine whether relative delay has an effect on a player's decision to join the server. Figures 5.11(a), 5.11(b) and 5.11(c) show the distribution of the nominal measures of relative delay, *stddev*, *avgratio* and *topratio* for tourist and regular players. Players with delays over the 95th percentile of 546ms have been removed.

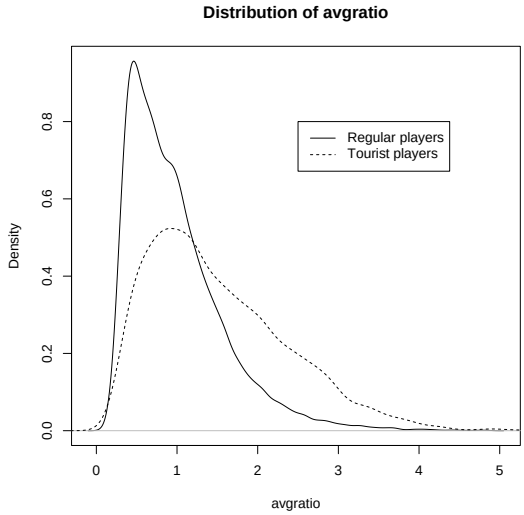
For *topratio* (Figure 5.11(b)), peaks can be seen towards 0 and at 1. This is because when there are a small number of players on the server, *topratio* will tend to 0 or 1 depending on



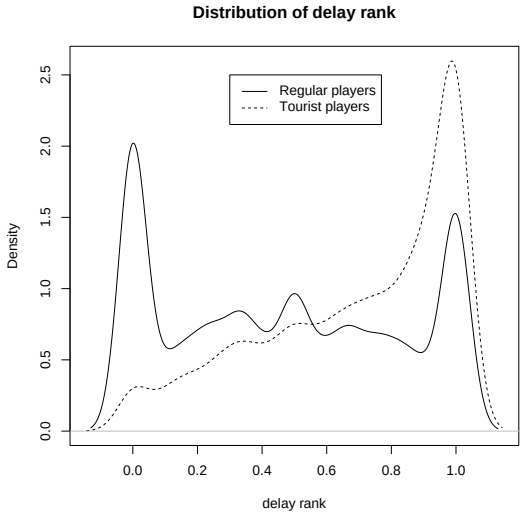
(a) *stddev* for regular and tourist players



(b) Distribution of *topratio* for regular and tourist players



(c) Distribution of *avgratio* for regular and tourist players



(d) Distribution of *rank* for regular and tourist players

Figure 5.11: Relative delay for regular and tourist players

whether the player has a high or low relative delay; when there are only two players on the server, then the player with the lower delay will have a $topratio$ of 1, whereas the players with the higher delay will have a $topratio \approx 0$.

To examine whether the distributions of the relative delay metrics differ between those players who choose to stay and those who leave, we use the Wilcoxon rank sum test, since Figures 5.11(a)-5.11(c), and the Shapiro-Wilk test, indicate that the distributions are not normal. All three nominal relative delay metrics are found to have significant differences between tourist and regular players. For *stddev*, the value of the Wilcoxon rank sum statistic $W = 43913156, p < 0.01$. For *topratio*, $W = 51489024, p < 0.01$, and for *avgratio*, $W = 39398308, p < 0.01$.

Figure 5.11(d) shows the delay ranks of both tourist and regular players. As with *topratio*, we see peaks towards 0 and 1. Tourist players rarely have a low rank — they tend to have higher delays than the other players on the server. Many regular players, on the other hand, have a lower delay compared with the other players. A Wilcoxon rank sum test shows that, as with the nominal relative delay measures, the difference in delay ranks between the two groups of players is significant, $W = 41384968, p < 0.01$.

5.4.2 Number of players

In Chapter 4 we showed that the number of players on a server is a consideration for users. To verify this on our own servers, we calculate the average number of players on the server in the first minute of each player's session. We then compare this value over the sessions where a player leaves within one minute, with those where a player remains on the server. As we might expect, players tend to leave when there are fewer players on the server, $t(5759) = 8.0981, p < 0.01$.

The number of players on a server might also affect users' tolerances for delay. If there are many players on a server, the excitement and interest that this can generate for the participants might lead them to tolerate higher levels of absolute and relative delay, since they are interested in the game and might prefer to remain in it. In other words, as more players join the server, players might tend to become less "fussy" about their delay, as the game gets more exciting. We define a player's **fussiness** as:

$$f(n) = d_{min}(n)/d_{max}(n) \tag{5.1}$$

where n is the number of players on the server, d_{min} is the minimum delay observed on the server, and d_{max} is the maximum delay.

To examine fussiness, we looked at our *Half-Life* game server over one month, and windowed the number of players on the server at five minute intervals. At each five minute interval, we calculate fussiness by counting the number of players on the server and compare this with the minimum delay observed by a player d_{min} and the maximum delay d_{max}.

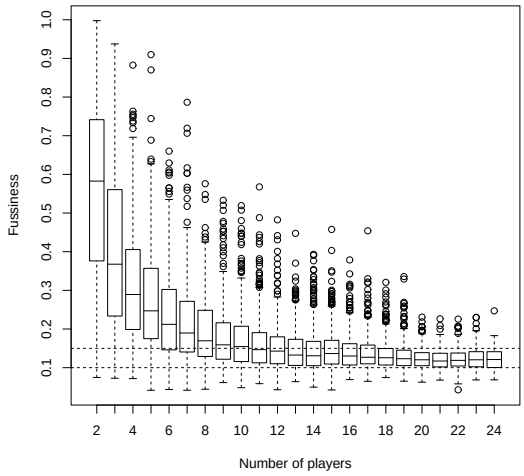


Figure 5.12: Fussiness versus the number of players on a server

Figure 5.12 shows a plot of the fussiness values, compared with the number of players on the server. It can be seen that fussiness tends towards 0.125. As the number of players on a server increases, it appears that users are willing to tolerate higher relative delays, up to a limit of eight times the lowest delay on the server.

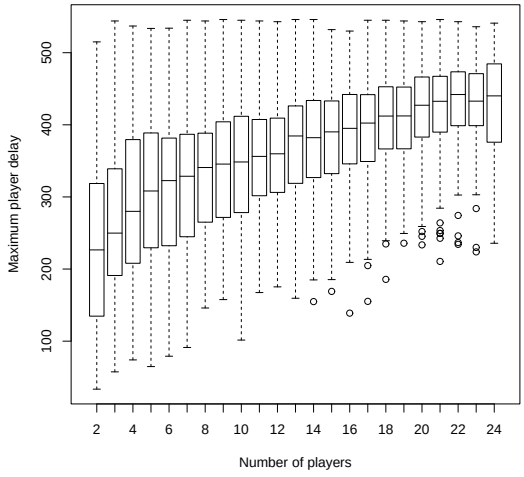


Figure 5.13: d_{max} versus the number of players on a server

As the number of players rises, the maximum of the delays observed by the connected players rises. The correlation between d_{max} and the number of players n is significant, $R^2 = 0.2176, p < 0.01$. It would appear that players with higher delay were more likely to stay when the number of players was higher — perhaps the increased enjoyment from playing with more

people offsets the deterimental effects of the network delay. To see if this is the case, for each unique player with an average delay greater than the average, we examine the connections where their delays are similar (within a 5% region), and compare the sessions where they choose to stay on the server with those where the choose to leave. The number of players on the server when players choose to stay is significantly higher, $t(213) = 5.6979, p < 0.01$.

5.5 Staying on a server

In this section we examine the network conditions that affect the length of a player's session. We might expect that with lower delay, players would tend to stay on a server longer, since lower delay would lead to a more enjoyable experience for the players, who would then wish to remain in the game for a longer period of time. Examining the relationship between the duration of a player's session and their average delay over a session, however, shows very little correlation: $R^2 = 0.03776, p < 0.01$ (Figure 5.14).

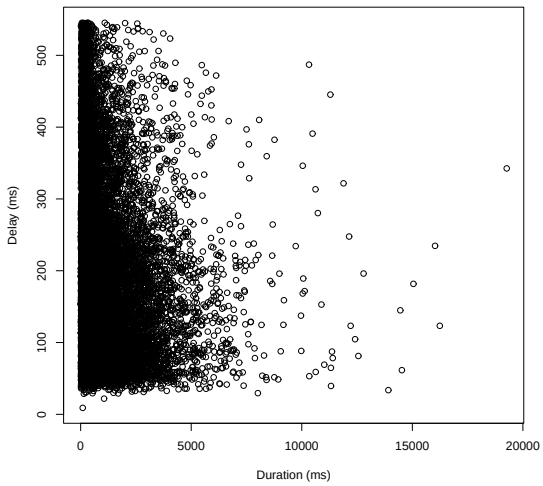


Figure 5.14: Players' average delay versus session duration

Since session duration is exponentially distributed, the tails, i.e., the sessions with extremely high durations, may obscure any relationship between duration and delay. Examining only those sessions with duration of less than one hour (Figure 5.15), however, also shows little correlation, $R^2 = 0.04591, p < 0.01$.

We were also unable to find any correlation between session duration and relative delay (Table 5.3). It would therefore appear that the length of time that a user plays on the server is not related to their network delay. To further analyse this we look at the conditions at the time when a player chooses to leave the server.

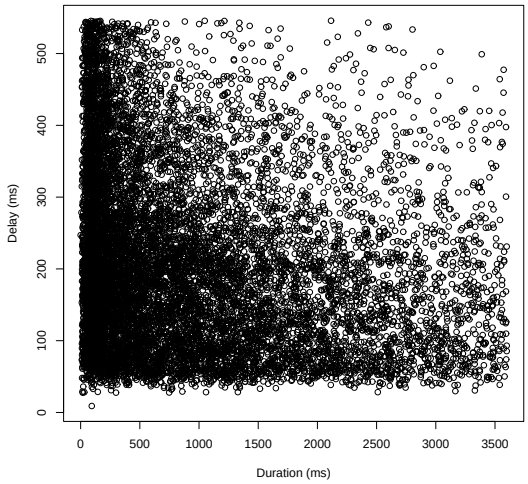


Figure 5.15: Players' average delay versus session duration where duration $\leq$ 60 min

Relative delay metric	R^2	p-**value**
stddev	0.03425	< 0.01
avgratio	0.0322	< 0.01
topratio	0.004156	< 0.01
rank	0.002881	< 0.01

Table 5.3: Relationship between relative delay and session duration

5.6 Leaving a server

In this section, we examine the network conditions that cause a player to leave a server. As we have seen that the individual player's level of delay has an effect on their decision to join a game server, we might expect that a player's delay would affect their decision to leave the server. A sudden increase in delay might lead a delay-sensitive player to become frustrated with the game, and perhaps to decide to disconnect and try and find another game server. Similarly, if relative delays are a concern for players, we would expect that new players joining the server with much lower delays than an existing player might lead that player, who would then have a much higher relative delay, to leave the server.

5.6.1 Number of players

As with joining a server, we examine the number of players on the server when a player leaves. Since we have observed that players tend to stay when there is a higher number of players, we might expect the converse to be true; that a decrease in the number of players might lead other players to leave. For each player's session of duration d, we calculate the average number of

players on the server in the first $d-1$ minutes of the session, and compare this with the average number of players in the last minute of a player's session. As expected, the number of players in the final minute is lower, $t(53822) = 73.5723, p < 0.01$. The difference between the number of players in the final minute and the rest of the session is very small (difference = 0.9072952), which affirms the findings of Section 4.4, where we noted that the correlation between a user's session duration and the number of players on the server was very small, but positive.

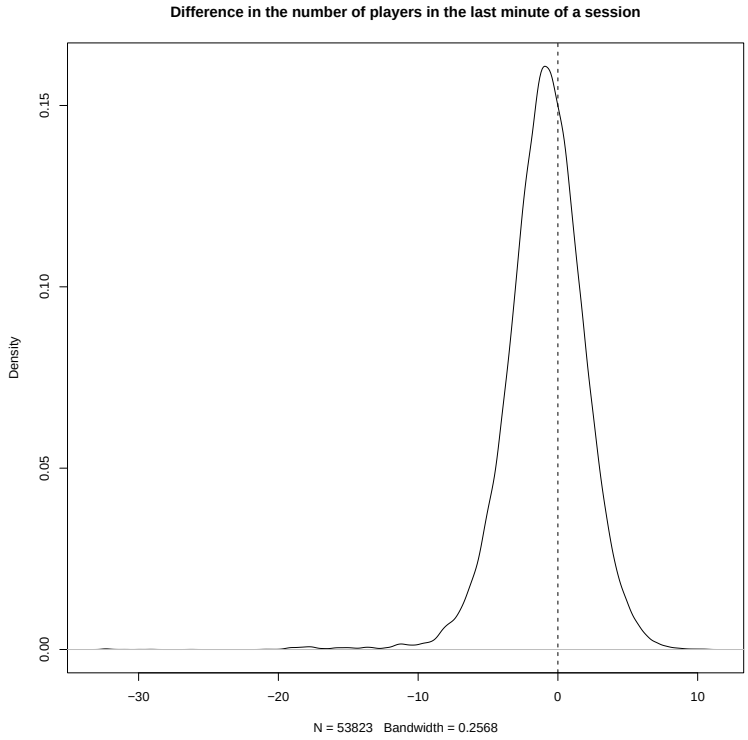


Figure 5.16: Number of players in the last minute of a session subtracted from the number of players in the rest of session

5.6.2 Absolute delay

To examine the effects of absolute delay on a player's decision to leave a server, for each player's session of duration d, where $d \geq 10$ min, we calculate the average delay in the first $d-1$ minutes of the session, and compare this with the average delay in the last minute of a player's session. If absolute delay has an adverse effect on a player's game and causes them to leave, then the delay in the last minute should be higher than that in the first $d-1$ minutes of the session. The average delay in the last minute, however, is significantly *lower*, difference = 8.514839, $t(51884) = 26.7843, p < 0.01$. This is the opposite of what was expected, and thus implies that delay, or as we had speculated, a sudden increase in delay, is not a determinant in a user's decision to leave a server.

To further examine the effects of delay on a user's decision to leave a server, an experiment was carried out on our two public game servers over one month. Every two hours, an additional level of delay was added to one of the servers for ten minutes. The other server had no additional delay, to act as a control. The exact timing of these additional delay periods varied randomly within a 20 minute time period, so that players would not notice a regularly occurring increase. The additional level of delay varied between 25 and 250 milliseconds — the upper bound of 250ms was chosen because this approximated the mean delay, and thus would be an increase of 100% for some players, which we assumed would be high enough to cause players to leave the server.

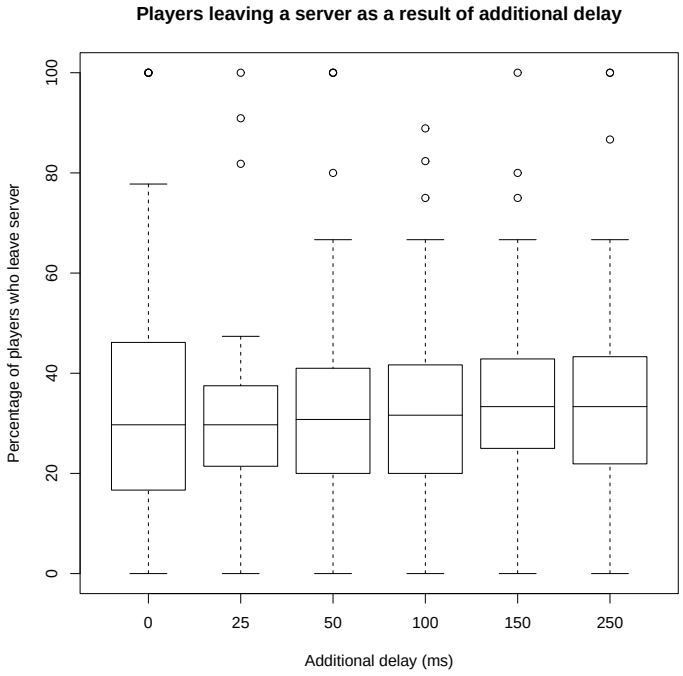


Figure 5.17: Players leaving a server as a result of additional delay

Figure 5.17 plots the percentage of players on the server that choose to leave in the ten minute period with added delay, against the level of additional delay (where 0 on the *x*-axis represents the control server with no additional delay). As the level of additional delay increases, there was little change in the percentage of players that chose to leave the server, which remained approximately 25-30%, even on the server with no additional delay.

The delay during the period with additional delay of the players who chose to leave the server (mean = 296.54ms) was significantly higher than that of those who chose to stay (mean = 250.34ms), $t(1507) = 3.7246, p < 0.01$. A rise in delay might therefore only lead a player to leave a game when the additional delay increases a player's delay beyond an absolute threshold.

We calculate *adddelay*, the increase in delay caused by the additional delay by considering a player's delay during a session without the additional delay, divided by the player's delay during the session with the additional delay. There was no significant difference in *adddelay* between those players who left ($adddelay = 0.7711$) and those players who stayed ($adddelay = 0.7735$), $t(2113) = 0.1065, p = 0.9152$, which indicates that a proportional increase in a player's delay has little effect.

Players who have played for a longer time might not want to leave the game, even in the presence of higher delay, since they may have been playing for a sufficiently long time to get engrossed in the virtual world. We examined the session duration prior to a period with additional delay of players who chose to stay and those who chose to leave. The duration of the players who chose to stay on the server was significantly higher, $t(3342) = 4.7339, p < 0.01$, with players who stayed having a mean duration of 2613.49 seconds, whilst players who chose to leave had a mean duration of 1796.54 seconds. The longer a player remains in the game, the more they might be enjoying the game, and hence the less likely they would want to leave, even in the event of additional network delay.

Regular players were no less likely to leave the server. The number of times a player had played on the server prior to an additional delay period was not significantly different between those who stayed (11.55) and those who left (10.87), $t(2895) = 1.12, p = 0.2628$.

One reason why players were not leaving the server in spite of the additional delay might be that they were unable to notice the delay. Analysing players' actions within the game indicates that this is unlikely, however. The average number of kills per minute made by players in periods with no additional delay was 1.430, which was significantly higher than the average of 0.6042 during the periods with additional delay, $t(3937) = 17.6115, p < 0.01$. The average number of times a player was killed per minute was 1.104 in the presence of additional delay, which was significant higher than the average of 0.0708 during the periods with no additional delay, $t(3467) = 67.499, p < 0.01$. The additional delay thus had a significant effect on players' performance, which we would expect they would notice. Although they could notice the delay and their performance was degraded, players were not inconvenienced to the extent that they would leave the server.

5.6.3 Relative delay

To examine whether relative delay affects a player's decision to leave a server, for each nominal relative delay metric, we look at the value in the last minute of a player's session ($metric_l$) divided by the value for the entirety of the session ($metric_e$). A result which differs from 1

should indicate that there is a change, which might imply that this is a factor in the user's decision to leave the game.

For each of the nominal relative delay metrics, there was a significant difference between $metric_l$ and $metric_e$. $avgratio_l$ differed from $avgratio_e$ by 0.03742295, $t(11650) = 12.9954, p < 0.01$. $topratio_l$ differed from $topratio_e$ by 0.06407042, $t(11650) = 32.9701, p < 0.01$. $stddev_l$ differed from $stddev_e$ by 9.728487, $t(11650) = 17.099, p < 0.01$. Although statistically significant, the difference in means between $metric_l$ and $metric_e$ are very small, as can be seen in Figures 5.18(a)-5.18(c).

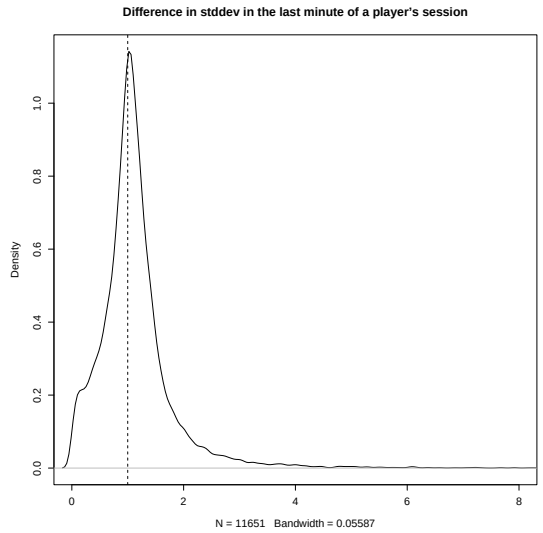
Relative delay *rank* is measured on a scale which includes 0, so dividing is impractical. Instead we subtract the *rank* in the last minute of a player's session from *rank* over a player's session. Figure 5.18(d) indicates that $rank_l$ is marginally higher than $rank_e$ — a paired t-test shows the differences in means is 0.0565336, $t(11650) = 23.299, p < 0.01$.

In Section 5.6.2 an experiment was carried out whereby delay was introduced to all the players on the server, to see whether this would cause them to leave the game. We could not conclude that additional delay caused players to leave the server. One reason for this, however, might be that all the players were experiencing additional delay, and therefore they might believe that this was a level playing field, since everyone was being affected by the delay. If only some players were experiencing additional delay, they might feel disadvantaged, and perhaps choose to leave the server.

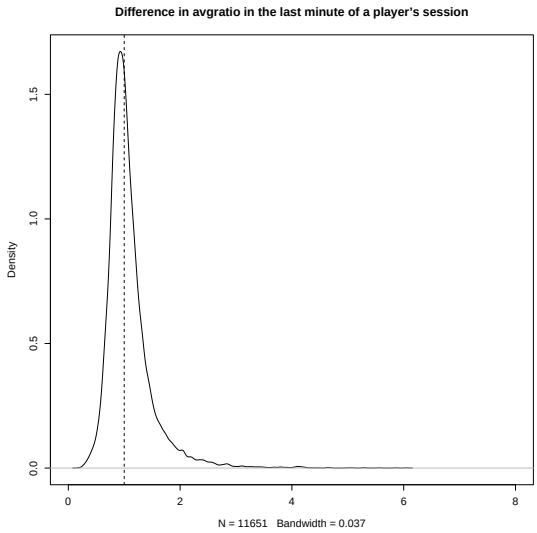
To determine whether players would leave if additional delay caused them to have higher relative delays, the experiment described in Section 5.6.2 was repeated, but instead of introducing additional delay to all the players, 20% of the players were randomly chosen to receive additional delay.

Figure 5.19 shows the percentage of players who received additional delay that left the server when this additional delay was introduced. The average percentage of players who left was 32.55%. In the experiment where all the players received additional delay, an average of 33.96% left the server (Figure 5.17). We cannot reject the hypothesis the means of these two measures are the same, $t(258) = 0.4858, p = 0.6275$. It would appear that an increase in relative delay is no more a component in a player's decision to leave a server than an increase in absolute delay.

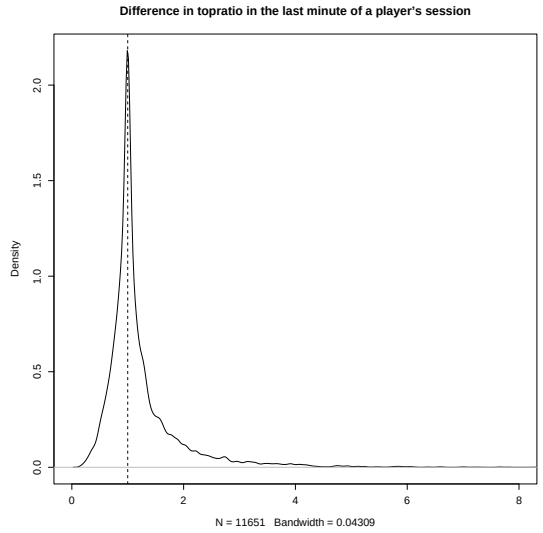
Players who had been playing for longer were again less likely to leave the server in the event of additional delay, $t(399) = 4.5723, p < 0.01$. Players who received additional delay and chose to stay had an average duration of 2234.102 seconds, whilst those who left had an average duration of 1304.488 seconds.



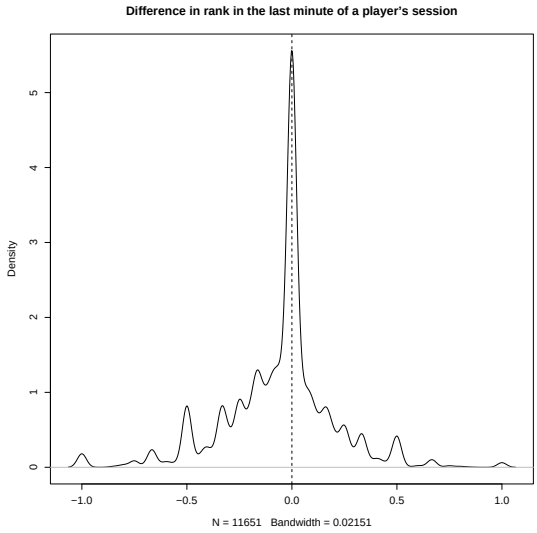
(a) *stddev* in the last minute of a session / *stddev*



(b) *avgratio* in the last minute of a session / *avgratio*



(c) *topratio* in the last minute of a session / *topratio*



(d) *rank* in the last minute of a session - *rank*

Figure 5.18: Relative delay effects in the last minute of a player's session

As with the addition of delay to all the players, players who received additional delay performed worse during these periods. The average number of kills per minute in the absence of additional delay was 1.456, as opposed to 0.6233 with additional delay, which is a significant difference, $t(3035) = 15.4988, p < 0.01$. The number of times that a player was killed also increased significantly under additional delay, from an average of 0.6042 times per minute to 1.430, $t(3937) = 17.6115, p < 0.01$.

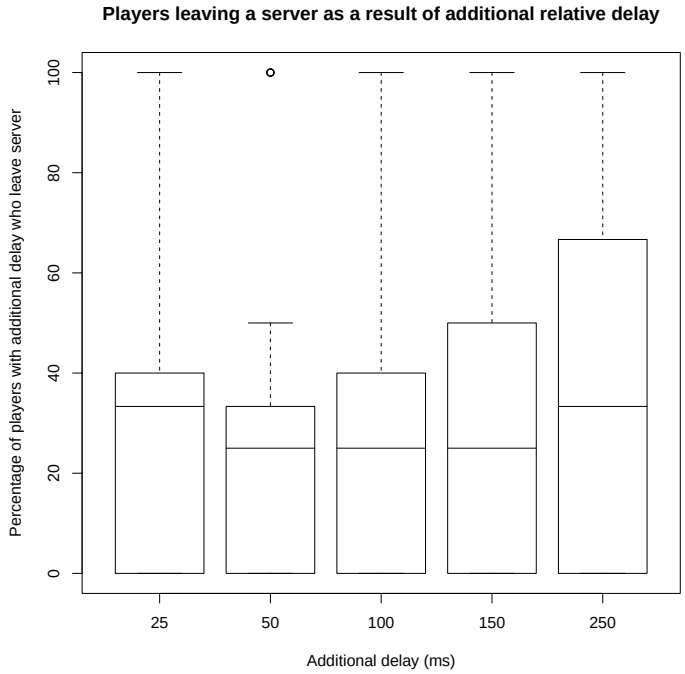


Figure 5.19: Players leaving a server as a result of additional relative delay

Adding delay to some of the players on the server created effects which were similar to those when delay was added to all of the players on the server. We are thus unable to conclude that additional relative delay causes a player to leave a server, in spite of the noticeable detrimental effects on player performance.

5.7 Summary

In this chapter we have analysed the delay characteristics of players connecting to publicly available *Half-Life* game servers. By examining the delays when a player chooses to join a server and leave a server, and by inserting additional delay into the path between a player and the game server, we have inferred users' preferences towards network delay.

In this chapter, we have indicated the following:

- Players on our servers have a mean absolute delay which is approximately 250ms
- Absolute delay is a determinant in a player's decision to join a server
- There is little relationship between delay and a player's session duration
- Absolute delay has little effect on a player's decision to leave a server
- Players who have been playing longer are less likely to leave in the event of additional absolute delay

- Relative delay has little effect on a player's decision to leave a server

It would appear that players are delay-sensitive in that they will not stay on a game server where they have a high absolute level of delay, or where their delay is higher than that of the other players on the server. Once the game is in progress, however, they become less delay-sensitive, and do not respond to increased delay. Perhaps as a result of the fact that players do not appear to leave a server as a result of network delay, we found that there was little correlation between a player's delay and the duration of their session.

Some of the effects that we have described and analysed in this chapter are not clearly significant. For instance, we found a difference in the relative delay between tourist and regular players that was significant, but this difference was very small. These unclear results may be a result of using publicly-available servers, where there are many additional factors that are outside our control. These uncontrollable factors may obscure results or affect experiments. In the next chapter, we conduct experiments under controlled conditions to further analyse game players' perceptions of absolute and relative delay.

Chapter 6

The effects of delay on FPS game players

Chapter 4 demonstrated that players consider the presence of other players when connecting to an FPS game server. In Chapter 5, we showed that users consider delay when connecting to a game server. They do not, however, seem to consider their relative delay, and we found little or no relationship between relative delay and the length of time that a player remains on a server, or the conditions that lead a player to leave a server. This is surprising, as we expected that users would consider the delay of the other players on a server.

The results in Chapter 5 came from publicly-accessible game servers. As with all measurements taken from the Internet, there are many uncontrollable variables that might affect an analysis. For instance, the hardware of the users that connect to these servers might differ, and this could have an effect on their perception of network latency. In this chapter, we explore users' preferences for absolute and relative delay in a controlled environment.

Two methodologies are used. A questionnaire was used to explore users' perceptions and preferences about delay. Secondly, a set of usability experiments was performed in a controlled environment. The aim of these experiments was to answer the following questions:

- **Q1** What level of delay do users notice?
- **Q2** Do users prefer similar or different delays to each other?
- **Q3** Do users perform differently under different levels of delay?

One reason for users preferring relatively similar delays is that this presents a "level playing field". If all the users have the same handicap resulting from network delay, then no user would have any advantage and so perhaps the network delay would be less relevant. If this is the case, then we would expect that game players' performance under different levels of network delay would be the same, if all the players had the same level of delay.

6.1 A survey of game players' perceptions of network conditions

6.1.1 Methodology

A questionnaire comprising 23 questions was designed. The questions spanned two areas of game design — the effect of delay on game players, and the effects of network disruption in a virtual world. Where appropriate, the questions involved answers on a Likert scale [127] of seven, and in addition respondents were invited to make any additional comments. The questions used in the questionnaire are listed in Appendix C.1.

The questionnaire was placed on a World Wide Web server, and advertised via game servers and various games-related mailing lists. 22 respondents were interviewed in person, and there were 236 unique responses via the website. Multiple responses from the same IP address within a six-hour period were assumed to be errors and were ignored.

6.1.2 Results

Three questions were used to determine respondents' experience and skill with networked games. These are depicted in Figures 6.1, 6.2 and 6.3.

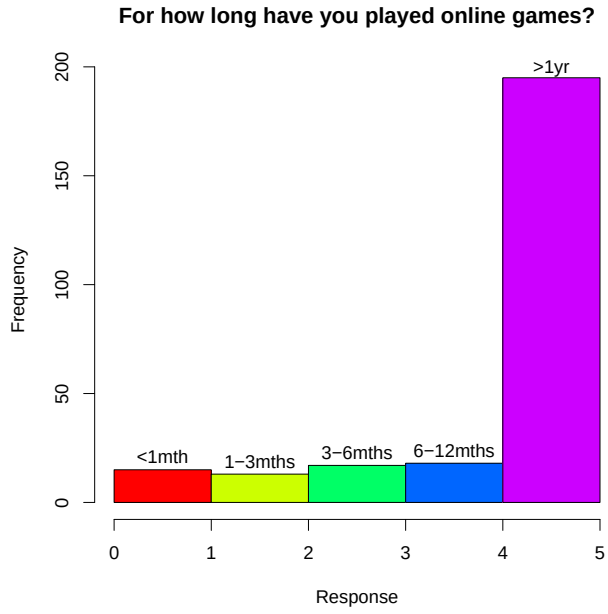


Figure 6.1: How long have respondents played networked games

As respondents were self-selecting (i.e., they chose to access the questionnaire of their own volition), this is not a random sample of the population. For instance, we would expect that most of our respondents play networked games, whereas Nie and Erbring's study of a more general population sample indicates that 35.5% of the population play games [150]. In comparison, only 5.81% of the respondents to our survey had played games for less than one

month, and 75.58% had played games for over one year (Figure 6.1). Thus we can expect this sample population to be representative of "expert" game players. This is appropriate since it is this class of player who is more likely to be concerned about network conditions and QoS. We classify the users who have played for over one year as "experienced" users, and use this as an independent factor for analysis of the answers to the other questions in the survey. Respondents were also asked to rate themselves as game players (Figure 6.3). The majority of respondents thought that they were better than average (median response = 5).

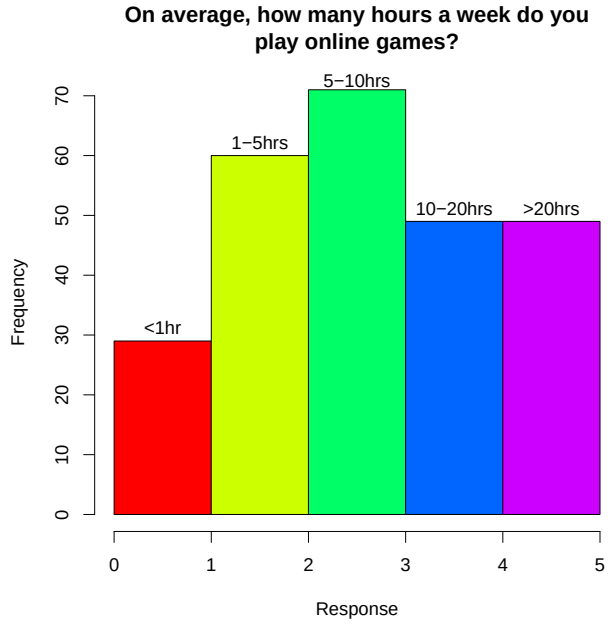


Figure 6.2: How often do respondents play networked games

Figure 6.2 shows that the amount of time that respondents play networked games is distributed across the spectrum (from a 5 point Likert scale, the interquartile range is 2). The median response was 3, or "5 – 10hrs per week". This is much higher than the amount observed in surveys of a more general user population. For instance, Swickert *et al.*'s study [191] found that users played games for an average of 60.52 minutes per week. This again indicates that the population of respondents is more interested in games, in contrast to surveys such as that of Hills and Argyle [92], where games are found to be the least popular networked application, and Aronsson *et al.* [12], where games are of below average interest for both commercial and residential Internet users.

Offering variable levels of QoS is difficult to implement if users do not suffer different costs for choosing between the levels of QoS. Respondents were asked two questions relating to their expenditure on playing games, which are presented in Figures 6.4 and 6.5.

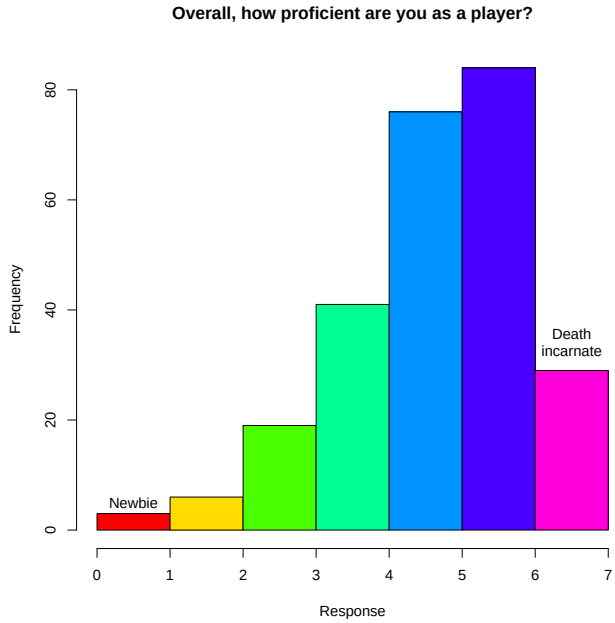


Figure 6.3: How well do the respondents think they play networked games

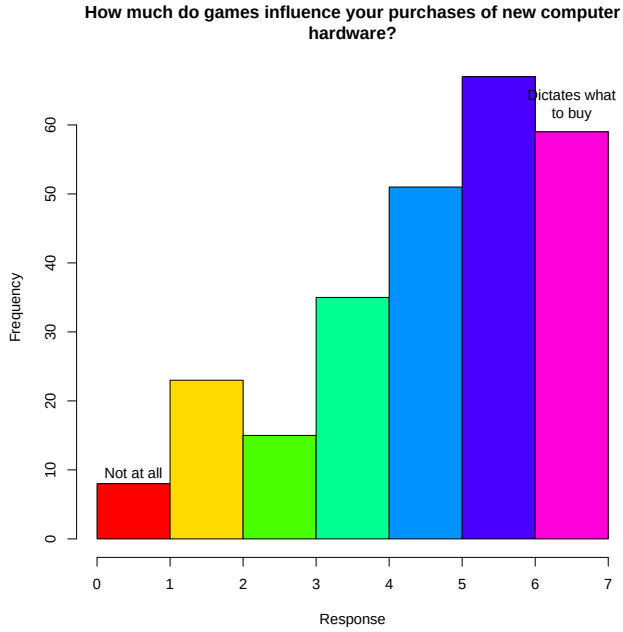


Figure 6.4: Do games affect respondents' expenditure?

Respondents were asked how much games influence their expenditure on computer equipment (Figure 6.4). Games were an important influence for the majority of respondents (82.17% of answers were between 4 and 7, with a median response of 5). To look at unexperienced versus experienced players, we use a Mann-Whitney test [182], as the *t*-test is inappropriate since the Likert scale used in our questionnaire does not have an equal interval scale. The Mann-Whitney

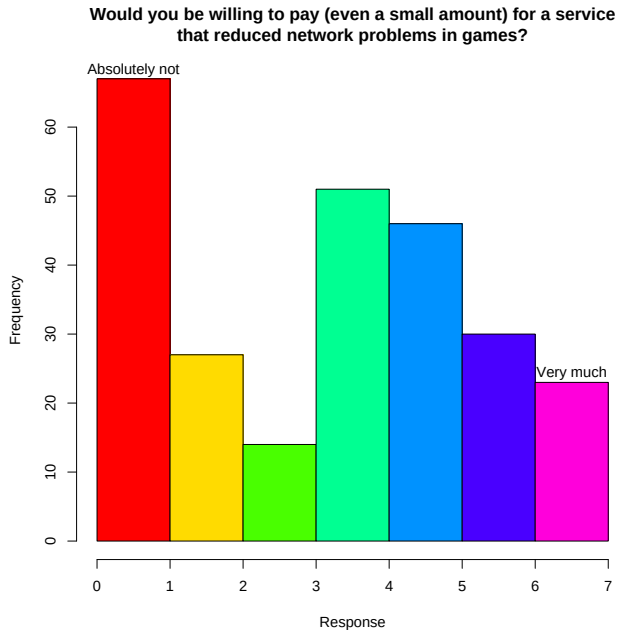


Figure 6.5: Are respondents willing to pay for QoS?

test indicates a statistically significant difference in the answers from experienced players, for whom games are a more important influence on their expenditure ($U = 7231.5, p < 0.01$). This indicates that the sample population is already spending money according to their playing of games, and therefore they might perhaps be interested in QoS, and the possibility of attaining better QoS through additional expenditure. When users were explicitly asked whether they would be willing to pay for better QoS, however, the responses were mixed (Figure 6.5, median response = 4, interquartile range = 4). There was an insignificant level of correlation between spending money on gaming through hardware purchases, and willingness to pay for QoS (correlation coefficient = 0.083069). Some of the comments from respondents indicate that additional payments for QoS might not be popular:

- "Couldn't someone else pay i.e. like the game developers, or maybe pay through advertising"
- "I'd like the ISP's [*sic*] to reimburse us for network problems"
- "Am willing to pay for a better connection, am using adsl but i refuse to pay extra online fees"
- "pay enough for my connection already"

To determine respondents' attitudes towards network delay, they were asked the five questions presented in Figures 6.6-6.11.

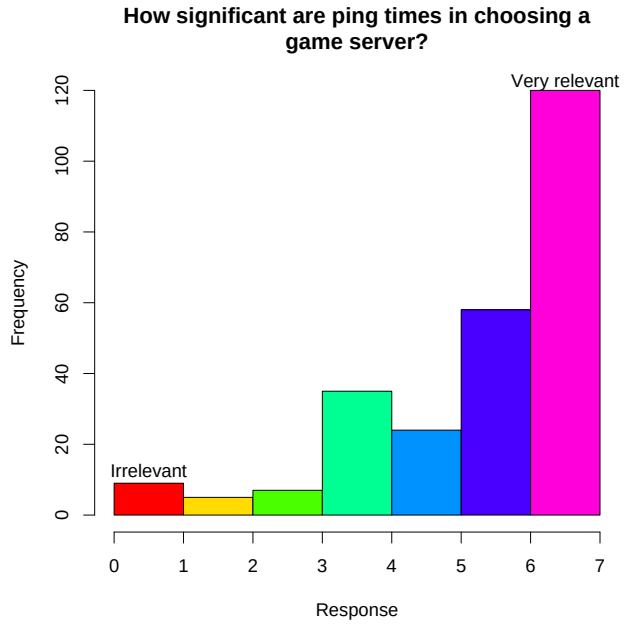


Figure 6.6: Do users consider delay when connecting to a server?

To determine respondents' preferences for absolute delay, users were asked whether they considered delay when connecting to a server (Figure 6.6). Delay was considered a very important factor by 46.51% (120) of the respondents, and the median response was 6. Delay was significantly more important for experienced players ($U = 6844.5, p < 0.01$).

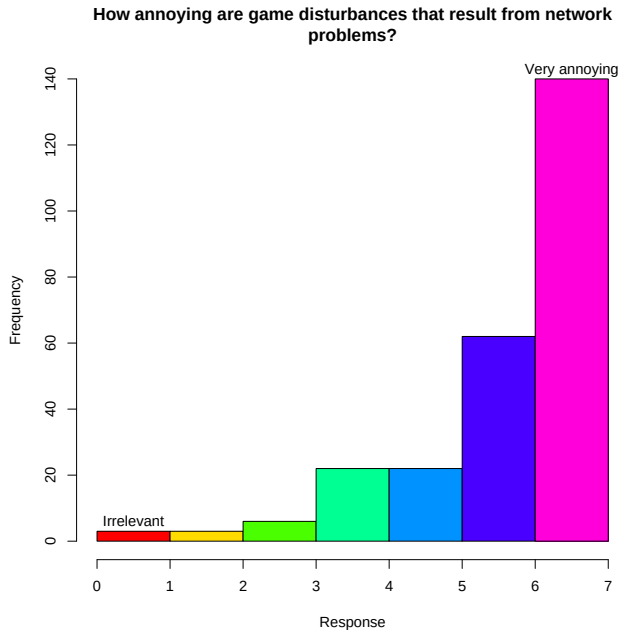


Figure 6.7: Do network problems annoy players?

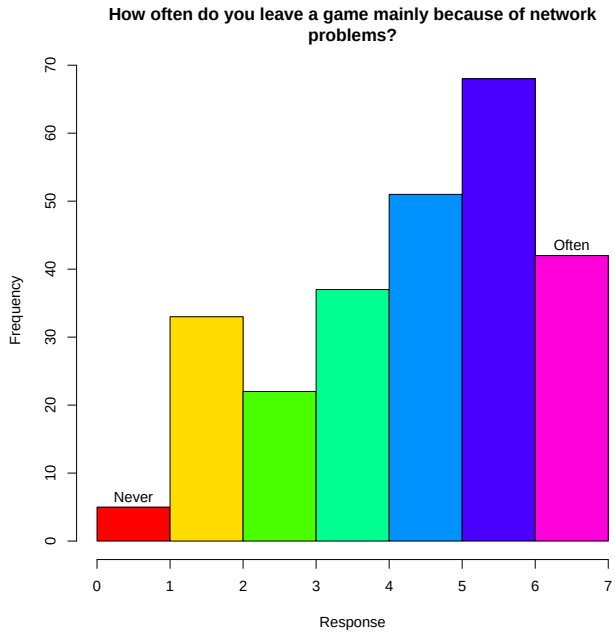


Figure 6.8: Do network problems lead players to leave a game?

Users overwhelmingly found network problems an irritant in playing games (Figure 6.7, median response = 7). When users were asked whether they left a server due to network problems (Figure 6.8), however, network problems seemed to be less important when leaving than when connecting to a server (median response = 5), and there was no significant difference between responses from experienced and inexperienced players ($U = 5921.5, p = 0.3533$).

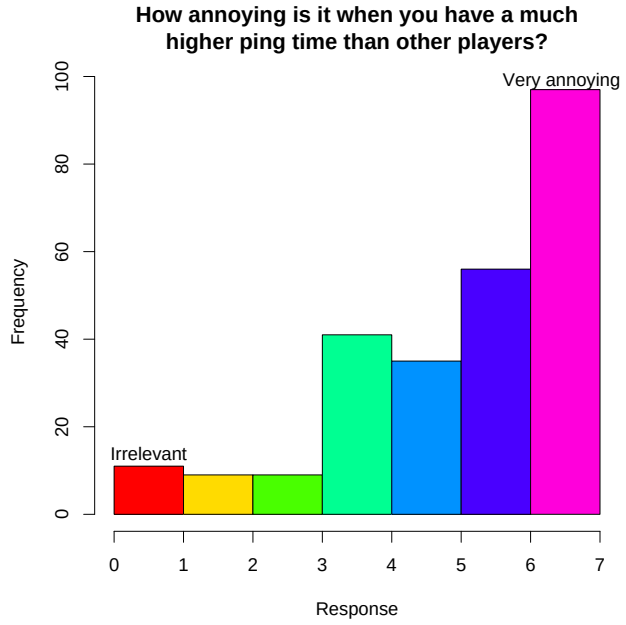


Figure 6.9: Do respondents like relative delay?

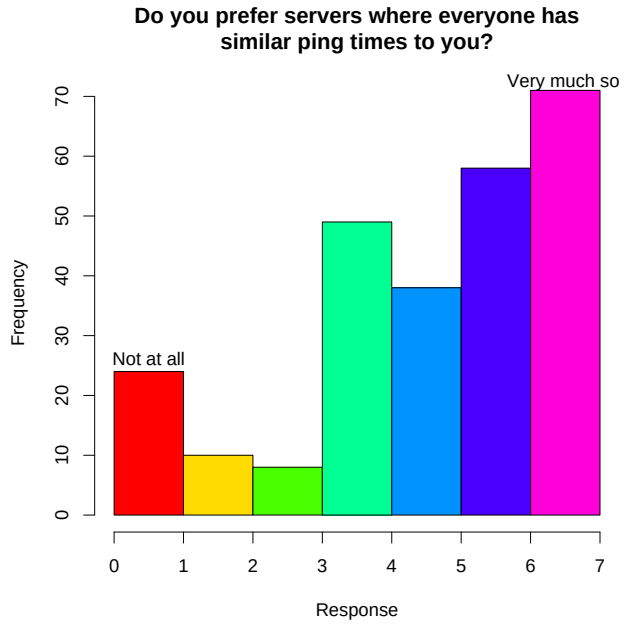


Figure 6.10: Do respondents prefer relative delay?

To determine respondents' preferences for relative delay, they were asked whether they found it annoying to have higher delay than the other players on a server (Figure 6.9), and whether they preferred to have the same delay as all of the other players (Figure 6.10). Players find it very annoying to have higher delays than other players (Figure 6.9, median response = 6), and this is true for both experienced and inexperienced players ($U = 6029, p = 0.2373$). Most players prefer servers where everyone has similar delays (Figure 6.10, median response = 5.5), and experienced players had slightly higher preferences for this ($U = 6875, p < 0.01$).

As described in Section 5.2.2, *Half-Life* and other FPS games typically offer players the ability to check their "ping", or application-level delay, via a scoreboard which shows the delays for all of the players. If game players are concerned about their delay, then we would expect them to check this value often. If users never check their delay, then it might be more difficult for them to gauge their relative delay compared with the other players, and might also indicate that they do not care about such relative delays. Respondents were asked how often they check their "ping" during the course of a games session (Figure 6.11). The results were mixed, with a median response of 4, and an interquartile range of 3. Experienced players tend to check their delay by a significantly higher amount ($U = 6808.5, p < 0.01$).

6.2 Game players' perceptions of network delay

The questionnaire indicates that game players are aware of, and are concerned about, network delay. A questionnaire, however, only indicates what players claim, and this might differ from

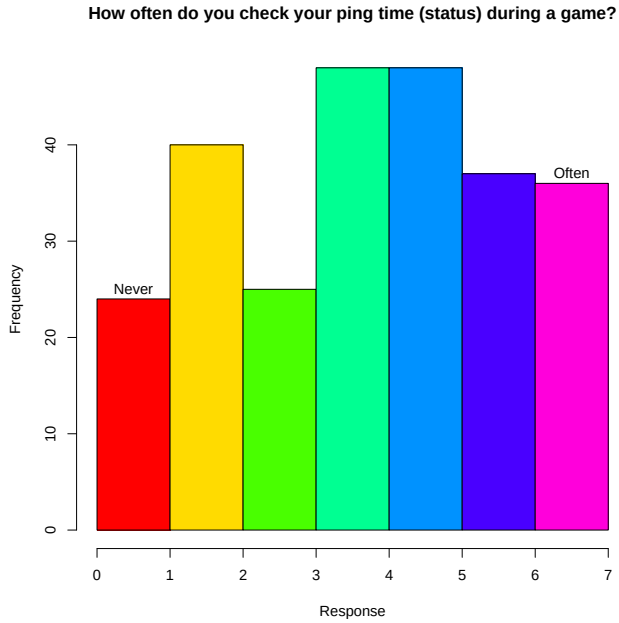


Figure 6.11: Do users check their delay during games?

how they actually behave in practice. In order to further examine users' perceptions of their absolute network delay, a set of experiments were carried out to to determine what level of delay could be noticed by game players.

6.2.1 Methodology

A *Half-Life* server identical to those used in Chapter 5 was connected to a *Half-Life* client machine via a gateway machine (Figure 6.12). The gateway machine, a dual-processor 2x200MHz Pentium Pro PC, running Linux 2.4.9, was used to introduce delay into the network in the same manner as the experiments described in Section 5.4, using the *iptables* and *libipq* interfaces in Linux. The client machine was a 733MHz Pentium III, running Windows 2000 and the *Half-Life* client version 1.1.0.8.

Delay was introduced in the network in the direction from the client to the server. This was deemed to be more appropriate since it meant that the game server program should have reacted to the clients as if they were users on high-latency links.

The task that each experimental subject was asked to perform was to aim and shoot using the "rpg_rocket" weapon. This weapon represents a rocket-propelled grenade launcher with a laser sight, and the position of the laser is calculated at the server (this laser sight is visible as the red dot that is circled in Figure 6.13). Any network latency can thus be observed by the user in the form of a delay between the user moving the weapon and the laser sight moving in response.

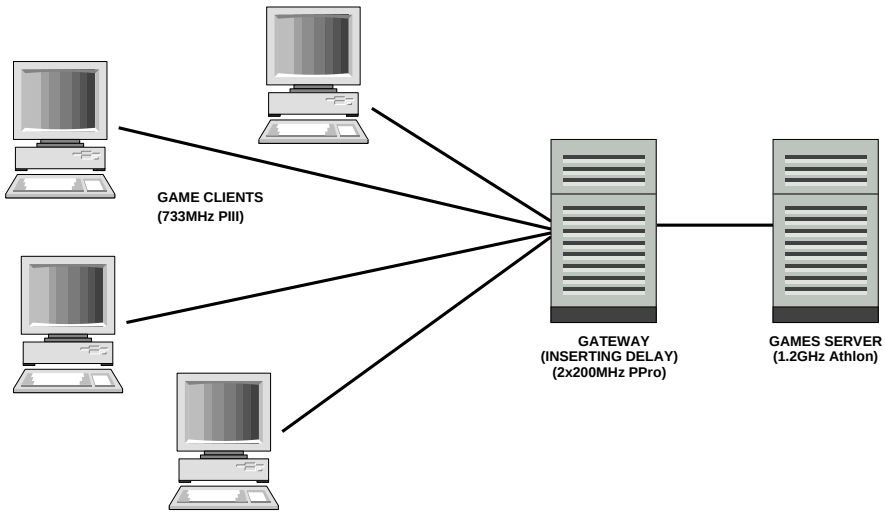


Figure 6.12: Experimental network setup

Figure 6.13: Monitoring the effect of network latency in *Half-Life* — the weapon's laser sight (the circled red dot) is calculated server-side

Participant Age Group	No. of males	No. of females	Total
20-35	5	3	8
36-49	7	2	9
$\geq$50	1	0	1

Table 6.1: Demographics of participants in single-player experiments

Usability tests of audio, video and other multimedia applications typically utilise the five-grade Mean Opinion Score (MOS) methodology [101]. Several problems have been found, however, with MOS and multimedia applications, and in particular networked multimedia applications [199]. To overcome some of these problems, the triangular taste-test method [35] was used to determine discriminability of delay. After an initial time period for participants to acclimatise to the game and environment, users were asked to complete the task, that is, to aim and shoot, three times. In one of these three tasks, delay was introduced between the client and server, and the task with additional delay was randomly selected. This set of three tries was repeated for a variety of levels of delay ranging between 50 and 750ms, and the order in which these different levels of delay were introduced was also random. After each set of three tries, users were asked which of the three they thought was the odd one out. In addition, they were asked to indicate on a 10cm line [190] how sure they were of this answer (the sheet which experimental subjects had to fill out is listed in Appendix C.2.1). This 10cm continuous line was used in favour of the discrete MOS five-point scale. Each session was recorded using the game's built-in recording facilities. Eighteen subjects participated in total. There were five female subjects, and the subjects ranged in age from 20 to 51, with a median age of 37.5 years (Table 6.1 offers a breakdown of the participants by age group). Six of the subjects claimed to have extensive experience with FPS games, whilst the others were either complete beginners or had played occasionally.

In Chapter 5 we observed that most regular players on the server had delays in the region of 50-300ms, whilst the 95% percentile was 546ms. This implies that users should be able to perceive delays in these regions. The experiments therefore introduced five different levels of delay for users to attempt to detect: 50, 150, 250, 500 and 750ms.

6.2.2 Results

Table 6.2 shows the subjects' answers to the question "which was the odd one out?" for each level of delay. We expect that by pure chance, a subject would choose the correct answer (that is, the session with additional introduced delay) $\frac{1}{3}$ of the time. The σ units represents the number

Delay (ms)	Percentage of correct selections		σ units
	Obtained	Expected	
50	44.44%	33.33%	1.0
150	66.67%	33.33%	3.0
250	77.78%	33.33%	4.0
500	88.89%	33.33%	5.0
750	83.33%	33.33%	4.5

Table 6.2: Triangular test results for perception of delay

Delay (ms)	χ^2	p-value
50	1.0	0.3173
150	9.0	0.0027
250	16.0	< 0.01
500	25.0	< 0.01
750	20.25	< 0.01

Table 6.3: χ^2 values for triangular test of perception of delay

of observed correct selections in excess of the expected number of correct selections that would occur by chance, and is calculated as [35]:

$$\sigma units = (n - N_p)/\sqrt{N_p q} \tag{6.1}$$

where:

n = number of correct choices

N = total number of users

p = probability of a correct choice by chance

q = probability of an incorrect choice by chance

A σ value which is ≥ 3.09 is considered very highly significant ($p < 0.001$). We can therefore conclude that users can notice a 250ms level of delay. At a lower level of significance ($p < 0.05$), we can conclude that users notice a 150ms level of delay. This is also indicated by the χ^2 test values (Table 6.3).

Figure 6.14 shows that users were more confident about distinguishing the presence of network delay when the delay was high. This is to be expected, since a higher level of network delay should be more intrusive in the game and thus more noticeable. A repeated-measures

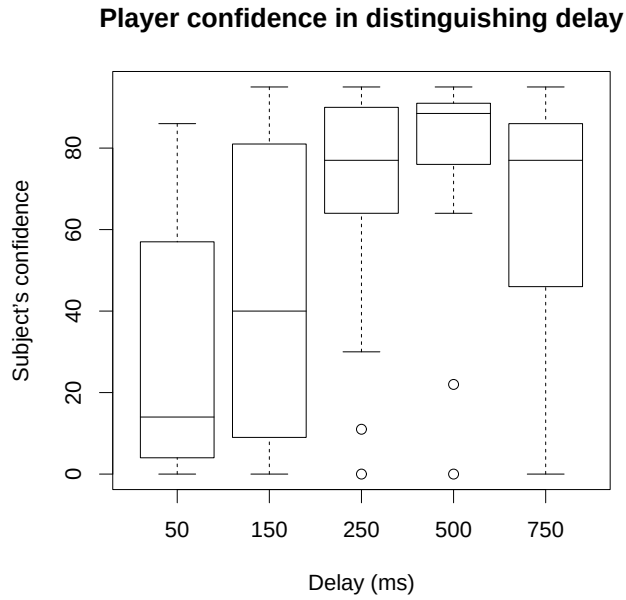


Figure 6.14: Users' confidence in distinguishing delay

Factor	**ANOVA test results**
Age	$F(13,2) = 0.3774, p = 0.8916$
Experience	$F(1,2) = 0.9389, p = 0.4348$
Sex	$F(1,2) = 0.2100, p = 0.6917$
Age:delay	$F(4,52) = 0.9623, p = 0.58052$
Experience:delay	$F(4,8) = 0.2889, p = 0.87723$
Sex:delay	$F(4,8) = 0.3781, p = 0.58153$

Table 6.4: ANOVA test results for perception of delay versus sex, age and experience.

ANOVA test between the level of delay and a user's confidence was statistically significant, $F(4,68) = 9.6866, p < 0.001$. Users' confidence that they were distinguishing network delay rises markedly at the 250ms level. The median confidence level at 150ms was 40, whereas at 250ms it was 77.

Repeated-measures ANOVA tests indicated no significant effect for age, sex or experience on a subject's ability to distinguish different levels of network delay. There was also no significant variation within any of these factors (Table 6.4). This result is surprising as one might expect that experienced players might be more capable of detecting network delay, since they would have been exposed to its effects in a gaming scenario more frequently.

6.3 Game players' perceptions of relative network delay

6.3.1 Methodology

In order to examine users' perceptions of their relative network delay, a multiplayer experiment was carried out involving groups of players. The relative delays that different members in the group experience were adjusted, and users' objective and subjective preferences were recorded. The *Half-Life* server used in Section 6.2 was connected to six identical client machines, comprising 733MHz Pentium III PCs running Windows 2000 and the *Half-Life* client version 1.1.0.8, connected via a 100BaseT switch.

These experiments were designed to provide controlled conditions, and so some changes were made from the game server set-up that was used for the servers placed on the public Internet. In particular, the game clients and server were configured so as to maintain similar conditions for all the players. We had observed from separate experiments [143] and discussions with game players that network conditions tend to be noticed when there is a great deal of interaction between the players. When a player is exploring the virtual world by themselves, there is less likelihood of an inconsistent state, and so server-side mechanisms such as dead reckoning can be effective and make any network delay less noticeable for the user. Therefore, for the purpose of these experiments, a small *Half-Life* map, "G0b-forsaken", was used, since some of the larger maps which are designed for more than 16 to 24 players, tend to have a correspondingly larger geographical area. Such a large map might mean that the smaller number of players used in these experiments might never interact with each other.

To maintain similar conditions for all the players, the server was configured in "Internet mode" (setting *sv_lan 0*), and client-side prediction and server-side delay compensation were disabled (setting *cl_pred_fraction 0* and *sv_unlag 0* respectively). The map was modified so that only one weapon, the "9mm assault rifle", was available for users, and each client was limited to the character avatar "Grunt". The in-game scoreboard was disabled by changing the standard key binding for displaying the scoreboard from the *Tab* key to the *Break* key (there seemed to be no other easy method of doing this without access to the game's source code, and in any case this method had the desired effect since the *Break* key does not appear to function during the game).

In order to isolate the effects of network delay, the experiments were designed to attempt to minimise disruptions caused by non-networked effects. Bichard [25] lists five non-networked ways in which players may become disrupted in an FPS game:

1. the often erratic behaviour of other players — the continual shifting of strategies and alliances and the relative experience of other players
2. personal strategies and weapon choices
3. session time-outs — usually 30 minutes before a map change
4. player migration — as teams and individuals shift from game to game
5. new maps — although there are a core of maps that are continually replayed

We attempted to minimise these problems as follows. Users were sorted into groups of five players according to their level of experience: no experience, some experience and experienced (that is, those who had considerable experience playing FPS games). The game map was modified so that only one weapon was available for players to use, and each client was configured to use the same character avatar in the game. Client-side prediction was disabled, as were the server-side lag compensation mechanisms [24]. Although the experiments took place on a LAN, the server was configured in "Internet mode". The in-game scoreboard was disabled — the scoreboard displays each user's application-level delay, and so this could affect the results since users would know what levels of delay were being added in each experimental session. Users were also situated such that they were unable to communicate with each other physically, or view the screens of other players, in case this had an effect on the results. Each session was recorded using the game's in-built recording facilities, and additional external video recordings were taken of the players using a video camera and microphone which was placed behind each player.

Scenario S-I	All players with no additional delay
Scenario S-II	All players with 100 ms delay
Scenario S-III	All players with 250 ms delay
Scenario S-IV	4 players with 250 ms delay, 1 player with no additional delay
Scenario S-V	1 player with 250 ms delay, 4 players with no additional delay

Table 6.5: Experimental scenarios in multiplayer experiments

Five users participated at a time, for five sets of five minute sessions. Each session varied in the level of delay according to the scenarios listed in Table 6.5. There were three experimental sessions where all the participants received the same level of delay (Scenario S-I – S-III). Scenario S-I had no additional delay and acted as a control. Scenarios S-II and S-III involved inserting the same level of delay for all the players, and were designed to examine users' preferences and performance under similar delays. Scenarios S-IV and S-V involved inserting a

Participant Age Group	No. of males	No. of females	Total
$\leq$18	5	0	5
19-35	14	4	18
36-49	9	1	10
$\geq$50	2	0	2

Table 6.6: Demographics of participants in multi-player experiments

different level of delay for one of the five participants, and were designed to examine preferences and performance under different relative delays.

As the experiments took place on a 100Mbps switched LAN, there was minimal network latency apart from that which was deliberately introduced. Figure 6.15 shows the application-level delay that was recorded by the game, compared with the level of additional delay that was introduced by the gateway machine. On average, the overhead incurred by the application stack and Ethernet propagation created application-level delays which were 15.95759ms higher than the introduced amount of delay.

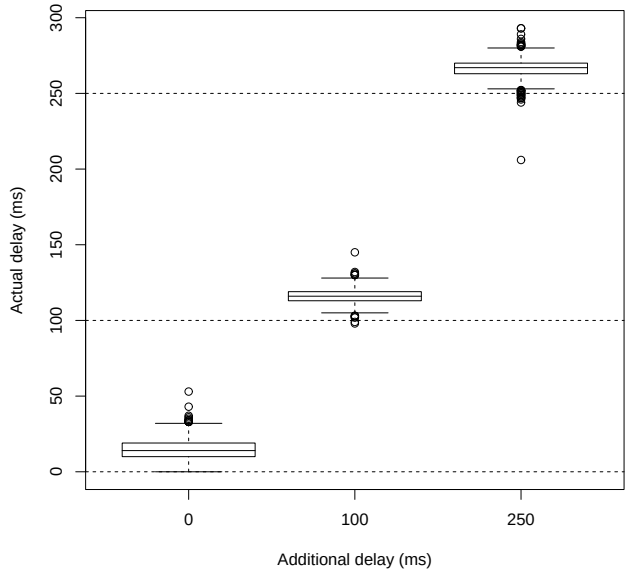


Figure 6.15: Application-level delay in multiplayer experiments

There were 35 participants, 30 male and 4 female, ranging in age from 9 to 55, with a median age of 32 years (Table 6.6 offers a breakdown of the participants by age group).

Users were given no particular task to perform. They were simply asked to "play the game" — this typically entails exploring the virtual world, shooting at the other players and picking

up weapons and ammunition. They were given five to ten minutes as a preparatory warm-up session, to allow them to get used to the environment and task.

6.3.2 Players' performance under delay

If delay is noticeable by a user then it might be reasonable to expect that a certain level of delay might also have an effect on a user's performance in the game. To measure performance we calculated the total number of times that a player killed another player (*kills*), and the number of times a player was killed by another player (*deaths*). We also examined the ratio of these two measures ($kills/deaths$) during each five minute session, since this might also reflect how well a player was performing if they adopted, for instance, a "kamikaze" strategy whereby they shot at everything in sight without caring whether they themselves were attacked.

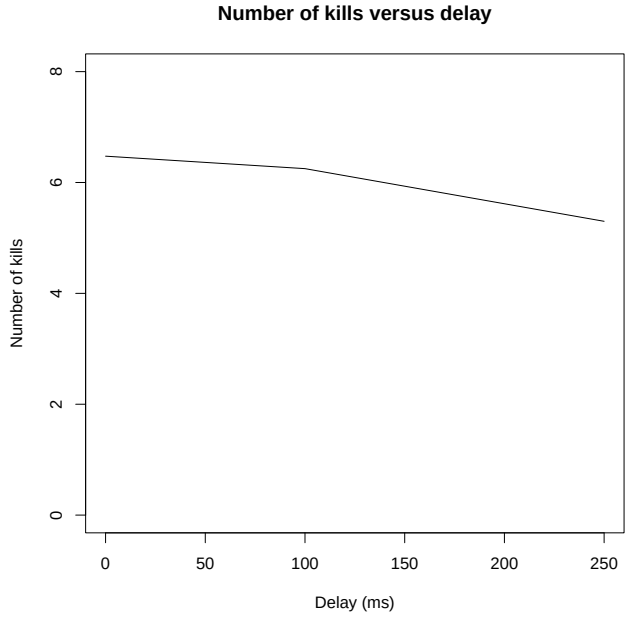


Figure 6.16: Kills versus delay

There is little change in player's performance under different levels of delay (Figures 6.16 – 6.18). A repeated-measures ANOVA test confirms that the number of kills a player makes is not related to delay, $F(2,68) = 0.6441, p = 0.5283$. The number of times a player is killed is also insignificant, $F(2,68) = 2.0379, p = 0.1382$, as is the ratio of $kills/deaths$, $F(2,68) = 0.4096, p = 0.6655$. The role of gender in video games has been examined, e.g. [119, 76], and games are often found to be male-dominated: "The aggressive nature of many games may, some fear, reinforce boys and leave girls behind in the field of computers, a field to which most children are first exposed by video games" [181]. FPS games in particular are singled out as being preferred by boys: "For most girls, the action in *Quake* is abhorrent and the plot

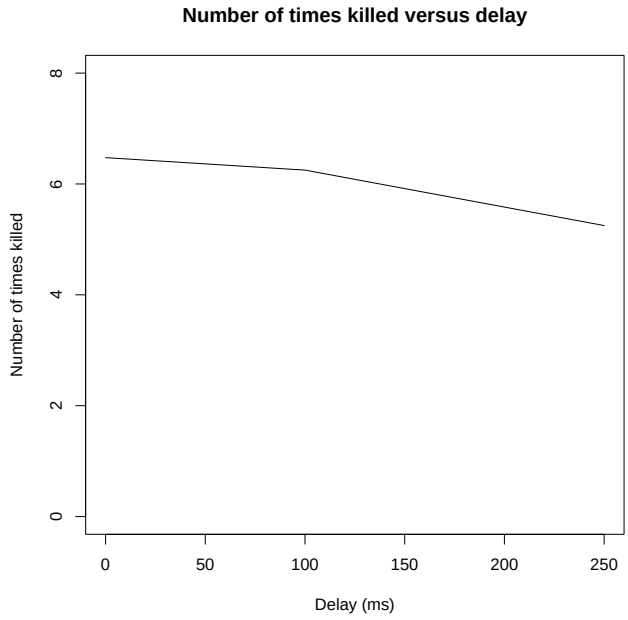


Figure 6.17: Deaths versus delay

nothing more than an exercise in aggression" [181]. It is therefore unsurprising that comparing a player's performance to their sex finds that male players killed significantly more players (a mean of 6.54) than female players (a mean of 2.88), $t(59) = 7.1183, p < 0.0001$. Male players were also killed fewer times (mean = 5.88) than female players (mean = 7.24), $t(37) = 2.61, p = 0.0129$. The small number of female subjects, however, means that these results are outside a 95% confidence interval.

Examining just the male players again found no relationship between the level of delay and performance. The number of *kills* was insignificantly related to the level of delay, $F(2,58) = 1.3347, p = 0.2712$, as was the number of *deaths*, $F(2,58) = 1.5271, p = 0.2258$.

To determine the effects of relative delay on performance, we examine the sessions with 250ms delay, Scenarios S-III – S-V. A dummy variable, $diffdelay$, was used to indicate whether a player had a different relative delay to the rest of the players. A $diffdelay$ value of 1 indicates that the player's delay was the same as all of the other players, whereas 0 indicates that the player's delay is lower, and a value of 2 indicates that the player's delay is higher than the other players. A repeated-measures ANOVA test indicates that *kill* is significantly related to $diffdelay$ at a 5% confidence level, $F(1,34) = 4.6364, p = 0.03848$. The number of *deaths* is also significant at the same confidence level, $F(1,34) = 4.4444, p = 0.04246$. $kills/deaths$ is not significantly related, $F(1,34) = 0.937, p = 0.3399$.

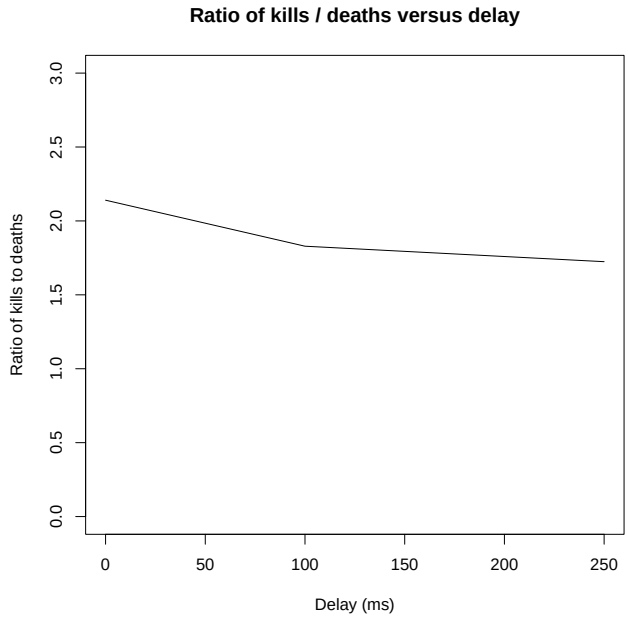


Figure 6.18: Kills over deaths versus delay

These results indicate that delay has little effect on a player's performance. In the uncontrolled wide-area experiments in Section 5.6.2, however, we found that the addition of delay did affect the performance of the players connected to the server. This difference may be due to the other variables that affected the Internet-based game servers. For instance, the additional delay might lead a "real" game player to disconnect from the server, whereas in our controlled experiments, this was not an option that was available. The players on the Internet-based game servers might also have been more varied in terms of their skill to play the game, and therefore their ability to cope with changes in network delay.

Players were asked in which session they performed the best. Each of the five sessions was ordered by each performance metric, *kills*, *deaths* and $kills/deaths$, to produce a rank in the range $\{1-5\}$. The distribution of the rank of the session in which users thought they performed best is depicted in Figure 6.19. It can be seen that this distribution is skewed towards the sessions with a rank of 1; most users accurately chose the session in which they performed best.

6.3.3 Players' enjoyment from a game

Users were asked how much they enjoyed each session. This was compared with how well a player performed in the session, to observe any correlation between performance and enjoyment. A within-subjects repeated-measures ANOVA test indicates that a player's number of

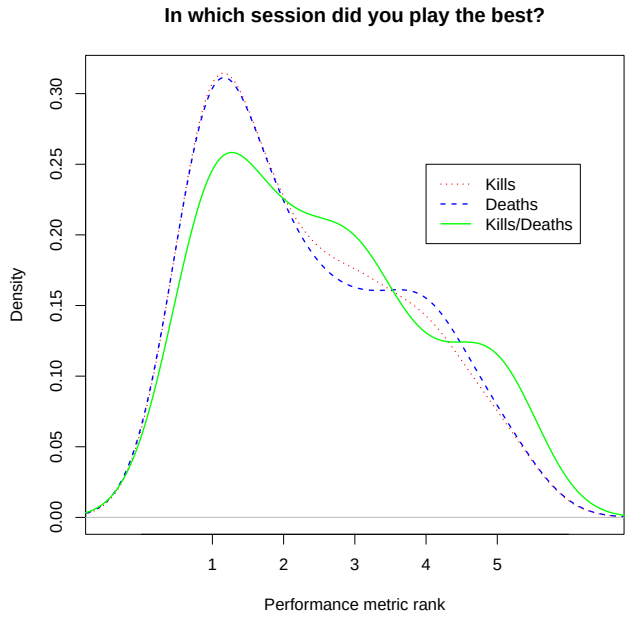


Figure 6.19: In which session did players think they performed the best?

kills is a significant factor in their enjoyment of a game, ($F(1,32) = 15.2125, p < 0.001$). The number of *deaths*, however, is insignificant, ($F(1,32) = 2.7515, p = 0.1069$).

As with performance, players were asked which session they enjoyed the most. The distribution of these answers, compared with how players performed, is shown in Figure 6.20. It can be seen that most players enjoyed the sessions in which they performed better. There was a high correlation between the answers to the two questions "Which session did you enjoy the best?" and "In which session did you think you played the best?" (0.4911782), which also indicates that a player enjoys a game more if they are playing well.

The level of network delay is also found to have a significant effect on a player's enjoyment, ($F(2,68) = 4.3411, p = 0.01681$). Even if all the players have the same high level of delay and thus a level playing field, the adverse effects to gameplay caused by the high delay, e.g., slow responsiveness, may create a less enjoyable experience for the player.

Relative delay seems to have little effect on a player's enjoyment. $diffdelay$ was unrelated to enjoyment, $F(2,68) = 0.458, p = 0.6345$.

6.3.4 Detecting delay

Subjects were asked four questions to see whether they could detect any difference between their latency and that of other players (Table 6.7). In addition, they were also asked whether they felt disadvantaged in a particular session, to determine whether players noticed, and were affected by, the lack of a level playing field.

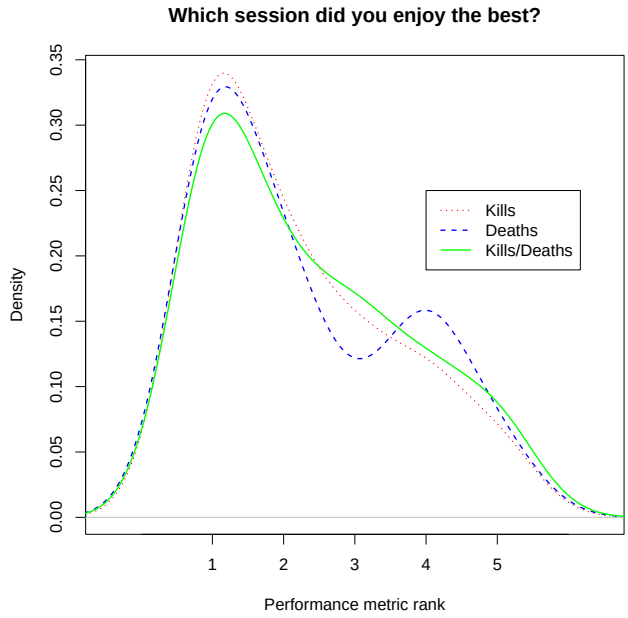


Figure 6.20: Which session did players think they enjoy the best?

DQ-1	Were you aware that other players were faster than you?
DQ-2	Were you aware that other players were slower than you?
DQ-3	Were you aware that you were faster than other players?
DQ-4	Were you aware that you were slower than other players?
DQ-5	Did you feel disadvantaged in that session?

Table 6.7: Questions about players' relative delay

Section 6.2 showed that game players could detect a 250ms level of delay. We would therefore expect that a player with a delay which is 250ms higher than that of the other players (i.e., experimental scenarios S-IV and S-V), would be detectable by the players. The same dummy variable $diffdelay$ was used as in Section 6.3.2. A within-subjects repeated-measures ANOVA test was conducted to see whether a subject's answers to the questions DQ-1 - DQ-4 varied with delay. $diffdelay$ was found to have no statistical significance in a subject's answer to DQ-1 ($F(2,68) = 0.5994, p = 0.552$), DQ-2 ($F(2,68) = 1.8663, p = 0.1625$) and DQ-4 ($F(2,68) = 0.5261, p = 0.5933$). $diffdelay$ was found to be statistically significant in a subject's answer to DQ-3 at a 5% confidence level ($F(2,68) = 3.6006, p = 0.03263$). The relationship between $diffdelay$ and a player feeling disadvantaged (DQ-5) was found to be statistically insignificant ($F(2,68) = 0.2823, p = 0.755$). We therefore cannot reject the hypothesis that users can detect delay when they have a lower delay than the other players on a

server, although they seem to be unable to detect delay when they have higher delays than the other players. They also seem to be unable to detect how other players' delays relate to their own.

Users may find it harder to distinguish the delay levels of other players, especially if the other players have different levels of delay (as is the case in sessions S-IV and S-V). This could explain the insignificant effects of $diffdelay$ on a player's answers to questions DQ-1 and DQ-2. They might find it easier to distinguish changes in their own delay, since they can more clearly observe differences in how the game reacts between sessions.

If game players can only correctly detect differences in delay when they have a lower delay than other players, then we would expect that relative delay would only have an effect on a player's enjoyment when this is the case. A repeated-measures ANOVA test of a player's enjoyment versus $diffdelay$ shows no significant relationship, $F(2,68) = 0.458, p = 0.6345$. Analysing only the sessions where players had a lower or the same delay as the other players showed no significant change in this relationship, $(F,1,34) = 0.6207, p = 0.4362$. We therefore are unable to conclude that a player's relative delay has any effect on their enjoyment of a game.

6.4 Summary

In this chapter we have examined how users believe that they respond to the presence of network delay, and compared this to how they actually react in laboratory conditions.

The results of the questionnaire indicate that game players, and in particular those players who have played games for a longer period of time, believe that delay is an important problem. Experienced players are concerned about both absolute and relative delay levels, and as a result they often check their delay level during a game.

The laboratory experiments, however, indicate that relative delay is not as important as the questionnaire had indicated. Players were able to detect an absolute level of 250ms in single-player tasks. Although they could detect this, raising their delay to this level did not decrease their performance in the game. It did decrease their enjoyment, and perhaps the noticeable disruption to the game caused by network delay is not enough to compensate for a level playing field.

Whilst absolute levels of delay were noticeable, in multiplayer tasks, players were unable to detect differences in their relative delay compared with the other players in the session, save when their delay was lower than that of the other players.

It is perhaps unsurprising that users can only detect when their delay is lower than that of the other players, and were unable to detect differences in relative delay under other conditions.

A player can easily verify their own delay between sessions, for instance by repeating a known task such as jumping or shooting, and comparing the response in different games. It might be more difficult, however, for them to gauge the delays of other players, since without control over these other players, they would be unable to observe these other players in situations that would be comparable between sessions.

These results can help to explain some of the behaviour that was observed in Chapter 5. We observed that the mean delay of all the players who connected to our game servers was 231.70ms, whilst the mean delay of those "regular" players who stayed for longer than ten minutes was 144.02ms. If players can notice a 250ms delay, then perhaps most players are choosing to ignore servers which have delays in excess of this figure. We also observed that relative delay had very slight effects, and there was a small, although significant, difference in relative delay between those players who stayed and those who left. Since most players were unable to detect differences in relative delay, except when they had a lower delay than the other players, the best way for players to gauge their relative delays is not through their own perceptions, but to use the scoreboard. The questionnaire, however, indicates that not all players check the scoreboard on a regular basis, and so perhaps this can help to explain the mixed effects of relative delay.

To summarise, in this chapter we have shown the following:

- Game players believe that both absolute and relative delay is important
- Players appear unwilling to pay for greater QoS
- Players can notice a 250 ms level of delay in *Half-Life*
- Lower delay does not increase players' performance
- Lower delay increases players' enjoyment of the game
- Relative delay has some effect on performance
- Relative delay has no effect on player's enjoyment
- Enjoyment of the game is related to how many kills a player carries out
- Players can only detect relative delay when they have a lower delay than other players

Chapter 7

Summary and conclusions

Networked latency is an important factor in the user's experience in a networked first person shooter game. This thesis has attempted to analyse whether a user's tolerance for the adverse effects of network delay is affected by the existence of the other players in the game. We originally presented the thesis that "users prefer similar relative delay, rather than minimal individual delay, in networked games, and will self-organise with respect to the other users in a game to achieve this." Although we have been unable to conclusively prove this thesis, we have analysed the thesis by examining the following questions:

- Do players in a multiplayer game consider the presence of other players when choosing where and when to play?
- Do players in a multiplayer game consider the network conditions of other players when choosing where and when to play?

The initial supposition for this work was that users in a multiuser application would consider the existence other users in that application, since some of their utility for using the application must be derived from these other users. Thus, we speculated that users might consider the network QoS that other users were receiving, and that the variation in a QoS parameter between the members of a group application might be an additional QoS parameter for such applications.

We have been able to conclude that users do indeed consider the delay of other players in some aspects of an FPS game. We have shown that **users consider other users when choosing to join an FPS game server**, and that **users consider absolute delay when joining an FPS game server**. The effects of relative delay on a player's decision to join or leave a game server were found to be insignificant, or very small. This does not mean that our original thesis was incorrect. We have been able to demonstrate that **users' behaviour towards delay changes over the course of a game**, and once a game in is progress, users can tolerate additional levels

of delay and do not leave the game. We have also shown that user's attitudes and perceptions of delay, and the effects of delay on their use of FPS games, differs from their measured behaviour.

In this chapter we outline the contributions of this thesis. We discuss how this relates to other research in this area, and suggest possible avenues for further work.

7.1 Contributions

In Chapter 4 we analysed users joining and leaving several FPS game servers on the Internet. The main contribution of this work is that, to our knowledge, this is the first large-scale session-level analysis of networked FPS games. The specific findings and contributions of this chapter were the following:

- We demonstrated that network externalities exist in FPS games, and hence that players consider the number of players in a game when choosing to join a server. This was shown through the high levels of autocorrelation in the number of players on an FPS game server in a time-series analysis. This is important since it means that we cannot treat a multiuser application such as a networked game in the same way as a single-source application, even though games might use unicast rather than IP multicast transport mechanisms.

- Players prefer to play at weekends and at particular times during the day. This was observed through the seasonal time-of-day effects in the number of players on a server, and is typically what one would expect for a recreational application such as a game.

- Player's session durations were random, whilst player interarrival times were less so. Analysis of duration and interarrivals found that players' session duration was exponentially distributed, and the distribution of player interarrival times were heavy-tailed.

In Chapters 5 and 6 we studied how players perceive delay in FPS games. We used detailed measurements of game servers that we ran on the public Internet, as well as usability tests in controlled conditions. Although many researchers have looked at delay-sensitive applications, and the delay requirements for multimedia applications, this is one of the first studies of networked FPS games in particular. The contributions of this work can be outlined as the following findings:

- Players consider absolute delay when choosing to join a game server. It appears that users' perceptions of delay change throughout a game, and delay does not affect a player's decision to leave a game server. Analysing the delay at the end of a player's session also failed to indicate that a rise in delay would cause them to leave the server. We further

tested this by adding delay to all the players on a server, and by adding delay to some of the players. Additional delay had no significant effect on the proportion of players who left the server, compared with those who left in the absence of additional delay. Players who had been playing for longer, however, were less likely to leave as a result of additional delay.

- Players believe that both absolute and relative delay are important factors in their gaming experience. They do not, however, appear to be willing to pay for lower delay or better network conditions.

- Players can only detect differences in relative delay when they have a lower delay than the other players in the FPS game *Half-Life*. They are able to notice a 250ms level of absolute delay, however.

- In controlled conditions, we found little relationship between network delay and a player's performance in, or enjoyment of, a game. Enjoyment does seem to relate to how many kills are carried out by a player. In practice, introducing delay into a public game server had an effect on player performance, which may be due to factors which were outside our control.

7.1.1 Discussion

This thesis has considered the FPS game as an example of a multiuser networked multimedia application. We speculated that an FPS game might differ from other multimedia applications such as video conferencing. A game is an entertainment application, and specifically an FPS game is an application for entertainment in a group scenario, where users compete against each other. Competition and entertainment are absent from a video conferencing scenario. Perhaps as a result of these differences, we have found that players consider the other users in a game, and that they are willing to tolerate quite high levels of delay.

We have conducted a number of measurements of publicly-available game servers. Our measurement methodology is **passive** — no additional traffic was inserted into the network, and no measurement software was required to be installed by game players. The choice of a passive methodology was motivated by the desire to make the measurements representative of "real" game players, and their behaviour in unadulterated game sessions. Our sampling may be less accurate than an active measurement methodology, but we believe that this is outweighed by the changes to user behaviour that may arise from the additional traffic and effort required to perform active measurements (both for the measurement site and the clients). Another benefit

of our passive methodology is that we have been able to monitor a much larger set of game servers and players than an active methodology would allow, since typically only a small subset of the overall userbase is willing to install additional software on their workstation for research purposes. In Chapter 5 we observed over 75,000 users connecting to our public game servers — it is highly unlikely that we would have been able to convince this number of users to install monitoring software. Moreover, since no additional software was required, users were unaware that they were being monitored, and so the behaviour that we have observed is more likely to be representative of normal user behaviour.

We have formulated a set of metrics to examine the variation in QoS between the participants of a multimedia session, and applied these to measurements of the players' network delay in an FPS game. Although we believe that the metrics are a valid indicator of the relative delays between players in a game, they may require further refinement, and this may be one reason for the size of the effects of relative delay that were observed.

We have found divergences between the results of the uncontrolled experiments (the measurements in Chapter 5 and the questionnaire in Chapter 6) and the controlled experiments (the usability experiments in Chapter 6). Users believe that relative delay is important, but in practice we found that absolute delay seemed to be more important in a player's decision to join a game server, and that in several cases users were unable to detect differences in relative delay between themselves and the other players. There was also a disparity in the relationship between performance and delay — in controlled experiments, we found that delay had little effect, but when delay was added to our publicly-available game servers, performance was degraded. These differences between theory and practice indicate that there are many variables that determine a player's decision to join or leave a game server. For instance, a player might leave a server because of the other players — they may have disagreements or arguments during the game, or have done so in the past. Alternatively, a player may leave because of a factor unrelated to the game or network — they may be hungry, or have an engagement which entails them leaving their computer or terminating their session. In order to address or control all of these variables, we need further information from the network and from the users.

These shortcomings in our methodology and metrics must be taken into account when considering the conclusions that we have presented. We lack perfect or complete information about the user population, and so uncontrolled variables may account for some of the phenomena that we have observed. Our controlled experiments, however, have managed to verify some of the results from the uncontrolled experiments, and we can be reasonably confident that users are able to detect levels of network delay, and that they consider this delay in a game.

7.2 Relationship to other work

Much of the previous work into multiuser applications has concentrated on IP multicast. The distributions for session duration and player interarrivals that we described in Chapter 4 appear similar to those that have been observed for some single-source multicast sessions [6]. If networked games are similar in some respects to multicast applications, this may be useful for future work, since it might be possible to use network and session traces and inferences from user behaviour in games to understand how to provide for multicast and multiuser applications.

The 250ms level of delay that we found to be noticeable in *Half-Life* is within the range that has been seen in other applications and human factors research. This confirms that networked FPS games are similar to other multimedia applications in terms of delay sensitivity. Networked games may therefore be suitable for further research into delay-sensitive applications and QoS. When examining "real" players on Internet game servers, however, we found that the mean delay experienced by players was in the 150-250ms range. The human factors evidence indicates that games should be unplayable above 200-250ms, whilst anecdotal reports indicate that games require delay bounds in the order of 100-150ms. We would therefore expect that 200-250ms would indicate the upper bound of the players' delay distribution, rather than the mean. It would appear that whilst players can notice a similar level of network delay in games to that in other applications, this does not deter users from playing to the extent that prior work would suggest.

We have been able to conclude that whilst users believe that their relative delay is important, the significance of relative delay in a player's decision to join a game is very slight. One explanation for this is that players are unable to notice when they have a different relative delay to other players, as we demonstrated in Chapter 6, except when they have a lower delay than the other players. Players are thus only able to gauge their relative delay by checking the scoreboard in the game, and our questionnaire shows that not all the players do so frequently during gameplay (Figure 6.11).

Another explanation for the minimal effects of relative delay is that whilst players believe that they are *altruistic*, in that they may leave a server where they have a much lower delay than the other players and choose to join another server where the playing field is level, in practice they may be *selfish* and only consider their own delay when choosing a game server. If a player happens to have a lower delay than the other players on the server, this is to that player's benefit, and so the player does not choose to leave. Similarly, if a player has a much higher delay than the other players, to the extent that the player's presence makes the game sluggish or irritating for the other players, the player with high delay acts selfishly and does not leave. Selfishness

is perhaps an unsurprising attribute to find in game players, given the competitive nature of the application. In Chapter 2 we described how choosing a game server with a low absolute network delay could be an instance of *rational* economic behaviour. There is also reason to believe from biology that an organism's degree of selfishness may be determined by their genetic similarities to other organisms [56]. People might therefore only act in an altruistic manner towards those to which they are related, or, "if each partner can get more out than he puts in" [56]. Hence, if there is no advantage to a player to leave a server where they have a different relative delay to the other players, they might not choose to do so.

7.3 Future work

This investigation into game players' perceptions of delay has resulted in many new questions and avenues for further work.

7.3.1 QoS, pricing and congestion control

We have chosen to analyse delay in FPS games because networked games are a real-time application of the type that QoS-enabled networks are designed to support. Any discussion of network QoS must involve pricing, since without price discrimination[1], network QoS cannot be provided, since any utility-maximising network user will always choose the highest available level of QoS if this is cost-free. One potential use for our work, and for future work derived from this thesis, is to help derive efficient pricing schemes and QoS policies for multiuser delay-sensitive applications.

If pricing is to be introduced for games, the charges should be acceptable to users. The questionnaire in Chapter 6 indicated that whilst players did not care much for paying extra for QoS, they were already indirectly paying to support their interest in computer gaming via purchases of new computer hardware. How to make the benefits of paying for better QoS an agreeable proposition for users is an open research question, for games and other multimedia applications. The success of subscription-based games such as *Everquest* indicates that one method is to charge for the content (access to servers in *Everquest*'s case), rather than for the network QoS itself.

In Chapter 4 we observed high variation in user duration and interarrival times, indicated by the exponential and heavy-tailed distributions. This has several implications for price stability and provisioning if the members of a game are to share the overall cost of a gaming session

[1] Price discrimination is a term from economics, defined as the situation where "two varieties of a commodity are sold (by the same seller) to two buyers at different *net* prices, the net price being the price (paid by the buyer) corrected for the cost associated with the product differentiation." [161]

amongst themselves. The autocorrelation in the number of players and interarrival times means that if the users are sharing the costs of a session, this cost will snowball; new users joining will be followed by other users joining (and users will join faster as the number of users increases), leading to rapid decreases in the cost per user, and vice versa for when users leave. This could be rectified, for example, by only changing the price for each user on a periodic basis rather than with each join or leave (we present a possible scheme in [88]). The autocorrelation seems to exist for large lags, however, which means that the periods of price reevaluation would also have to be large, and this could impede the efficiency of any pricing scheme.

Time-of-day pricing is commonly used for pricing utilities such as electricity and telephone service, and has been proposed as a simple, if suboptimal, method for pricing Internet traffic [130]. The time-of-day effects that we observed in Chapter 4 suggest that this might be appropriate on a per-application basis, at least for games. This might also have implications for network provisioning, whereby a network designed for games would have to be able to deal with the peak times.

We have discussed how games are useful for research into multicast applications because of their similarity at the session level. Current multicast charging proposals have tended to look at single-source applications, e.g. splitting the costs of a single-source application amongst the downstream members [90, 63], or by treating the multicast price as a proportion of the cost of the corresponding unicast (single-source) application [45, 168]. Most networked games tend to be multiple-source, since they typically involve more than one player. Data about user behaviour and preferences in networked games could perhaps be used to aid the derivation of pricing methods for multiple-source multiuser applications.

The presence of network externalities also has implications for congestion control. Legout *et al.* [120] suggest that the proportionally-fair bandwidth share for a multiuser application should relate either linearly or logarithmically to the number of downstream receivers. If network externalities do exist in multiuser applications, then a logarithmic relationship, using individual utility functions that incorporate network externalities, might be more appropriate. One way to encourage the use of bandwidth-saving techniques such as multicast for multiuser applications might be to offer a logarithmically-increasing bandwidth share to these applications, since users will be attracted to the application both by the network externalities, which encourage group usage, and the higher quality made available by the increased bandwidth share.

We have only examined one parameter of network QoS, namely, delay. In Section 2.2.1 we listed some of the other QoS parameters such as throughput and delay jitter. A future area of research could thus be how game players perceive and react to different absolute and

relative levels of these other QoS parameters. Although we have argued that delay is the most important QoS parameter for games, this does not mean that the other QoS parameters are irrelevant. Jitter has been found to be a problem for delay-sensitive NVEs [158], and whilst we have noted that throughput is not a problem for existing networked games, this may change in the future as higher bandwidth connections become more prevalent amongst residential game players. Further work is therefore required in this area.

7.3.2 Other games and applications

This work has only looked at one type of game, the first person shooter. The FPS game was chosen because it represents a popular networked game genre, and is amenable to large-scale experiments. Not all games, however, might respond to high network latency in the same way, and other games may have different delay requirements. Users of other games may thus perceive and react to delay in dissimilar ways to those that we have discussed here. Driving games may be even more delay-sensitive than FPS games [156], whilst it would appear that the RTS game is more tolerant towards high delays [192].

We have only looked at one set of FPS game servers in detail, namely, a set of *Half-Life* game servers that we installed at UCL. One reason for this was that we found it difficult to gain permission to monitor and/or install game servers at other remote sites. It might be useful, however, to conduct further experiments using a more widely-dispersed sample of servers. We were unable to explain why players connected to our UK-based servers from as far away as Taiwan, and monitoring servers around the world might help to determine the reasons for this.

Even within the same game genre, other FPS games might exhibit different characteristics to *Half-Life*. We have speculated that game players act selfishly in choosing a game server, and consider their absolute rather than relative delay. This might not be the case in team-based games, where players might prefer to have the same delay as everyone on their team. Altruism can sometimes take place if there are rewards to be gained over time [166], and this might perhaps occur amongst the members of a team in an FPS game, for instance to earn a reputation amongst the other members of the team. It might also be more appropriate in some team-based games if everyone has a similar delay, for instance if they are attempting to complete a shared task that requires synchronised input from each team member.

If, as the results of our questionnaire have shown, users in FPS games believe that they prefer similar relative delays to the other players on a game server, then this might be an appropriate QoS policy to offer to such game players. As well as being acceptable to players, it would also be useful for game design. The adaptive pipeline cheat-proofing protocol [49]

requires players to measure their delay to each other — this would be unnecessary if all the players knew that they had the same relative delays.

Games have been a popular application on NTT DoCoMo's mobile telephone network [174], and are viewed as an important source of revenue for the US and European 2.5G and 3G mobile networks [21, 153], with manufacturers creating handsets that are designed specifically for games rather than voice [152]. Current mobile games have tended to be single-player, and those multi-player games that do exist, such as *BotFighters* [113], are non-realtime, using mechanisms such as SMS (Short Messaging Service). As such, there have been few studies of the effects of network delay in realtime multiplayer mobile games. It is unclear whether our results will apply to mobile games — the different type of device or interface may mean that users have different expectations, or perhaps the problems caused by delay may manifest itself in different ways. On the other hand, some mobile games may entail users connecting a laptop or another familiar device to a wireless network [69], in which case we may expect users to respond to delay in similar ways to those that we have analysed in this thesis.

Apart from games, there are many other applications which require low levels of absolute and relative delay. For instance, online auctions might require that each bidder has similar delays to the auctioneer, so that one user could not use a lower delay to gain an advantage by knowing the value of bids before the other bidders. Networked gambling and online casinos are already a popular application on the Internet. These also require similar relative delays, for instance for each player in an online poker game, otherwise a player could again exploit a lower delay to observe another player's hand before making a bet. A QoS policy which provided similar delays to all the participants of a multiple-source application could therefore have a wide range of uses.

The data that has been collected and presented in this thesis has other potential uses for game design, besides the analysis of player delay. For instance, in Section 5.3 we noted that of the game servers that were advertised by the master server, only a relatively small proportion ($\approx 30\%$) were actually found to be reachable. This indicates that either the master server needs improvement, or perhaps that a better method for discovering game servers is required. We are currently investigating such a system, using a distributed peer-to-peer mechanism to allow users to discover the existence of game servers by querying players with whom they have recently played games [86].

Appendix A

ARIMA modelling

ARIMA (Autoregressive Integrated Moving Average) models were first introduced by Box and Jenkins [32], and have become a popular means of modelling time series data. An autoregressive process is defined as a serially dependent process whereby elements in a time series can be described in terms of previous elements:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \phi_3 X_{t-3} + \ldots + \varepsilon \tag{A.1}$$

A moving average process is where each element in a time series is affected by past errors, independent of the autoregressive process:

$$X_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \theta_3 \varepsilon_{t-3} - \ldots \tag{A.2}$$

An ARIMA model incorporates both the autoregressive and moving average processes. Such models are referred to as ARIMA(p,d,q), where p is the autoregressive parameter, d the number of differencing passes required to make the input series stationary, and q the moving average parameter. If the time series has a seasonal component, additional seasonal parameters are required, and the model is referred to as an ARIMA$(p,d,q) \times (P,D,Q)_s$ model, where P, D and Q repesent the ARIMA parameters of the seasonal component, and s is the period of the seasonality.

A.1 Diagnostic checking

An overview of diagnostic checking for periodic models such as ARIMA can be found in [139]. To check that a model fits the data, the residuals from the data are calculated and compared to white noise. The Ljung-Box "portmanteau" test [128] can be used to indicate the residuals' departure from white noise.

Appendix B

The FPS game *Half-Life*

This thesis has examined the FPS game *Half-Life*, for the reasons outlined in Chapter 3. This appendix outlines some of the details of the game and the protocols used to study *Half-Life* game servers.

B.1 *Half-Life* network protocols

As *Half-Life* is a closed-source commercial application, exact details of the game's networking code are unknown. Some details have been discovered, however, through network-level analysis, and can be found in [65, 143].

Both the client and server send generally constant bitrate streams. Client to server flows are generated at a constant rate of 24 packets per second, and over 99% of the packets are under 100 bytes in size. Server to client flows are generated at a slightly-varying rate of 16-17 packets per second, and vary in size, with a mean of 240 bytes.

All data communication uses UDP. Packets have sequence numbers for detecting loss and duplicates, and in certain cases, packets may be retransmitted. No congestion control is utilised, however, and retransmissions are at the same constant bitrate.

B.2 *Half-Life* query protocol details

Part of the data presented in this thesis is a result of querying several remote FPS game servers. To carry out these queries, we initially used a freely-available tool, *QStat* [165]. *QStat* was not ideal for long-term automated querying, and moreover was not as comprehensive a tool as we required. Thus, we wrote our own tool in Perl to query *Half-Life* servers in particular.

B.2.1 Master server query mechanisms

The "master server" acts as a central directory of game servers for game players to query when they wish to play a game.

- When a game server starts running, it registers with one or more server directories. The addresses of these directories are discovered by the server operator out-of-band and the server program is configured appropriately.
- A potential game player sends a 6 byte UDP packet containing the character “e”, followed by a sequence number indicating where in the list the server directory should start returning addresses. For the initial query, this number should be 0.
- The server directory responds with a UDP packet of up to 1396 bytes long. This contains a one byte sequence number, followed by a list of server addresses.
- If the sequence number returned by the server directory is non-zero, the client sends another query packet to the server directory, this time including the returned sequence number. The server directory then responds with another list of server addresses, until the client has received the entire list.

B.2.2 Server query mechanisms

Query string	**Type**	**Queried variable**
“info”	c	Information about the server (the name and description of the server and the name of the current map)
“details”	m	”info”, plus further details about the game server (server version, OS, the maximum number of players etc.)
“rules”	E	The rules of the game server (some rules are fixed by the game developers, but others can be configured on a per-server basis, e.g., whether it is password-protected, the length of time that each map is played for, etc.)
“ping”	j	Doesn’t return any data — used to determine the application-level delay to the server
“players”	D	Details about the players on the server (their name, number of kills and the length of time that they have been playing)

Table B.1: *Half-Life* server query variables

The game server itself offers a variety of configuration variables and statistics that can be queried from external hosts. These are all queried by sending the server a UDP packet containing a string which represents the variable that is to be queried. After receiving a query

packet, the server will return one or more UDP packets, containing one byte indicating the type of packet, followed by the requested information. Some of the strings that can be sent as queries are listed in Table B.1.

Appendix C

Questionnaires

C.1 Player survey

Please answer each question by placing a cross (X) in the appropriate box

1. For how long have you played online games?

<1mth	*1-3mths*	*3-6mths*	*6-12mths*	*>1yr*

2. On average, how many hours a week do you play online games?

<1hr	*1-5hrs*	*5-10hrs*	*10-20hrs*	*>20hrs*

3. How much do games influence your purchases of new computer hardware?

Not at all ☐☐☐☐☐☐☐ *Dictates what to buy*

4. Overall, how proficient are you as a player?

Newbie ☐☐☐☐☐☐☐ *Death incarnate*

5. When you are playing a game, to what extent are you aware of your surroundings (i.e., the world outside the computer)?

Not at all ☐☐☐☐☐☐☐ *Very much*

6. How much do you have a sense of being in the game world?

Not at all ☐☐☐☐☐☐☐ *Very much*

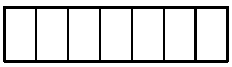
7. Do you have a sense of being in the same space with other players?

Not at all ☐☐☐☐☐☐☐ *Very much*

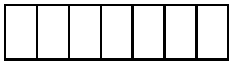
8. How often do you notice disruptions in the game (excluding external disruptions such as telephone, people interrupting, etc)?

Not at all  *All the time*

9. What proportion of game disruptions do you think are due to network problems as opposed to software problems?

Mostly software  *Mostly network*

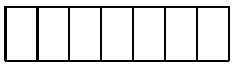
10. How annoying are game disturbances that result from network problems?

Irrelevant  *Very annoying*

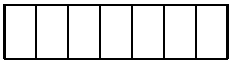
11. Do you become more aware of your physical surroundings when network problems occur?

Not at all  *Very much*

12. Do network problems disrupt your sense of being in the same space with other players?

Not at all  *Very much so*

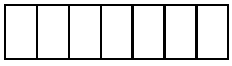
13. How often do you leave a game mainly because of network problems?

Never  *Often*

14. How significant are ping times in choosing a game server?

Irrelevant  *Very relevant*

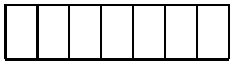
15. How annoying is it when you have a much higher ping time than other players?

Irrelevant  *Very annoying*

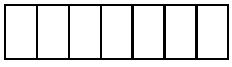
16. How often do you check your ping time (status) during a game?

Never  *Often*

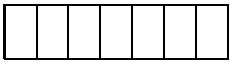
17. Do you prefer servers where everyone has similar ping times to you?

Not at all  *Very much so*

18. How often do you play games with people you already know?

Rarely  *All the time*

19. How often do you meet new players in games and play with them in future sessions?

Rarely  *Frequently*

20. Can you adjust your game play in the presence of network problems?

Not at all ☐☐☐☐☐☐☐ *All the time*

21. Does learning to anticipate network problems affect your game play?

Not at all ☐☐☐☐☐☐☐ *Very much so*

22. When network problems occur, how would you prefer to know about them?

☐ *Do not need to know*
☐ *Dialog box*
☐ *Flashing icon*
☐ *Integrated into gameplay*
☐ *Other:*

23. Would you be willing to pay (even a small amount) for a service that reduced network problems in games?

Absolutely not ☐☐☐☐☐☐☐ *Very much*

Any other comments?

C.2 Experimental questionnaires

C.2.1 Single-player experimental questionnaire

Name: _____

Sex: **M / F** **Age:** _____

Which task did you think was the odd one out?

1	2	3

How sure are you?

Not sure at all ——— Very sure

Which task did you think was the odd one out?

1	2	3

How sure are you?

Not sure at all ——— Very sure

Which task did you think was the odd one out?

1	2	3

How sure are you?

Not sure at all ——— Very sure

Which task did you think was the odd one out?

1	2	3

How sure are you?

Not sure at all ——— Very sure

Which task did you think was the odd one out?

1	2	3

How sure are you?

Not sure at all ——— Very sure

C.2.2 Multiplayer experimental questionnaire

Name: _____

Sex: **M / F** **Age:** _____

Please answer each question by placing a mark on the appropriate line

SESSION __:

Did you enjoy playing in that session?

Did not enjoy ——————————— Enjoyed

Did you feel disadvantaged in that session?

Disadvantaged ——————————— Advantaged

Were you aware that other players were faster than you?

Not aware ——————————— Highly aware

Were you aware that other players were slower than you?

Not aware ——————————— Highly aware

Were you aware that you were faster than other players?

Not aware ——————————— Highly aware

Were you aware that you were slower than other players?

Not aware ——————————— Highly aware

OVERALL:

Which session did you enjoy the best?

1	*2*	*3*	*4*	*5*

In which session did you think you played the best?

1	*2*	*3*	*4*	*5*

References

[1] 3Com. 3Com Scores With Modem Geared Towards Serious Gamers, Oct. 25, 1999. [cited 26 May 2002; 11:26 BST] Available from Internet: <http://www.3com.com/corpinfo/en_US/pressbox/press_release.jsp?INFO_ID=7138>.

[2] R. Adobbati, A. N. Marshall, A. Scholer, S. Tejada, G. Kaminka, S. Schaffer, and C. Sollitto. GameBots: A 3D Virtual World Test-Bed for Multi-Agent Research. In *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, Montreal, Canada, May 2001.

[3] D. Ahl. The Beginnings of Computer Games. *Computer Museum Report*, 22:3–5, Spring 1988. [cited 26 May 2002; 13:31 BST] Available from Internet: <http://ed-thelen.org/comp-hist/TheCompMusRep/TCMR-V22.html>.

[4] The All-Seeing Eye. [cited 18 Oct. 2002; 13:06 BST] Available from Internet: <http://www.udpsoft.com/eye/>.

[5] E. A. Alluisi. The Development of Technology for Collective Training: SIMNET, a Case History. *Human Factors*, 33(3):343–62, June 1991.

[6] K. C. Almeroth and M. H. Ammar. Collecting and Modeling the Join/Leave Behavior of Multicast Group Members in the MBone. In *Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing (HPDC-5)*, pages 209–216, Syracuse, NY, USA, Aug. 1996.

[7] C. A. Anderson and M. Morrow. Competitive Aggression Without Interaction: Effects of Competitive Versus Cooperative Instructions on Aggressive Behavior in Video Games. *Personality and Social Psychology Bulletin*, 21(10):1020–1030, Oct. 1995.

[8] J. Anderson. Who Really Invented the Video Game? *Video & Arcade Games*, 1(1), Spring 1983.

[9] J. Andreoni and J. K. Scholz. An Econometric Analysis of Charitable Giving with Interdependent Preferences. *Economic Inquiry*, 36(3):410–428, July 1998.

[10] G. Armitage. Sensitivity of Quake3 Players To Network Latency. In *ACM SIGCOMM Internet Measurement Workshop 2001*, Berkeley, CA, USA, Nov. 2001. poster, [cited 28 May 2002; 13:25 BST] Available from Internet: <http://opax.swin.edu.au/~garmitage/q3/imw2001/poster110101.pdf>.

[11] J. Aronson. Dead Reckoning: Latency Hiding for Networked Games. *Gamasutra*, Sept. 19, 1997. [cited 28 May 2002; 16:11 BST] Available from Internet: <http://www.gamasutra.com/features/19970919/aronson_01.htm>.

[12] M. Aronsson, D. Tholén, P.-E. Josephson, H. Li, and S. Kong. Broadband services for residential and commercial tenants: a categorisation of current and future services and a survey on tenants needs in Sweden. *Building and Environment*, 38(2):347–358, Feb. 2003.

[13] T. Aronsson, S. Blomquist, and H. Sacklén. Identifying Interdependent Behavior in an Empirical Model of Labor Supply. *Journal of Applied Econometrics*, 14(6):607–626, Nov./Dec. 1999.

[14] Associated Press. Gamers drive souped-up PC market. *CNN.com*, Aug. 26, 2002. [cited 23 Sept. 2002; 14:09 BST] Available from Internet: <`http://www.cnn.com/2002/TECH/ptech/08/26/hot.rod.computing.ap/index.html`>.

[15] R. W. Bailey. *Human Performance Engineering — Using Human Factors/Ergonomics to Achieve Computer System Usability*. Prentice Hall, Englewood Cliffs, NJ, USA, second edition, 1989.

[16] R. A. Bangun and E. Dutkiewicz. Modelling multi-player games traffic. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)*, pages 228–233, Las Vegas, NV, USA, Mar. 2000.

[17] R. A. Bangun, E. Dutkiewicz, and G. J. Anido. An Analysis of Multi-Player Network Games Traffic. In *Proceedings of the 1999 International Workshop on Multimedia Signal Processing*, pages 3–8, Copenhagen, Denmark, Sept. 1999.

[18] R. Bartle. Mud, Mud, Glorious Mud. *Micro Adventurer*, 1(11):22–25, Sept. 1984. [cited 26 May 2002; 13:32 BST] Available from Internet: <`http://www.mud.co.uk/richard/masep84.htm`>.

[19] D. Bauer, S. Rooney, and P. Scotton. Network Infrastructure for Massively Distributed Games. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 36–43, Braunschweig, Germany, Apr. 2002.

[20] N. Baughman and B. N. Levine. Cheat-Proof Playout for Centralized and Distributed Online Games. In *Proceedings of the 20th IEEE Conference on Computer Communications (INFOCOM)*, volume 1, pages 104–113, Anchorage, AK, USA, Apr. 2001.

[21] D. S. Bennahum. Be Here Now. *Wired*, 9(11), Nov. 2001.

[22] E. J. Berglund and D. R. Cheriton. Amaze: A multiplayer computer game. *IEEE Software*, 2(3):30–39, May 1985.

[23] B. Berkowitz. Happy Meals, Pentiums coming to video game world. *Reuters*, Sept. 16, 2002. [cited 17 Sept. 2002; 08:53 BST] Available from Internet: <`http://investor.cnet.com/investor/news/newsitem/0-9900-1028-20419348-0.html`>.

[24] Y. W. Bernier. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. In *Proceedings of the 15th Games Developers Conference*, San Jose, CA, USA, Mar. 2001. [cited 26 May 2002; 13:33 BST] Available from Internet: <`http://www.gdconf.com/archives/proceedings/2001/bernier.doc`>.

[25] J. P. Bichard <john@LIQUID-IDEA.COM>. Re: [GAMESNETWORK] The Medium That Has Not Yet Spoken Its Name, Sept. 10, 2002. E-mail to Games Research Network mailing list <GAMESNETWORK@liaani.UTA.FI>.

[26] S. Blake, D. L. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, Dec. 1998. RFC 2475.

[27] B. Blau, C. E. Hughes, M. J. Moshell, and C. Lisle. Networked virtual environments. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, pages 157–160, Cambridge, MA, USA, June 1992.

[28] N. S. Blomquist. Interdependent behavior and the effect of taxes. *Journal of Public Economics*, 51(2):211–218, June 1993.

[29] J. Blow. A look at latency in networked games. *Game Developer*, 5(7):28–40, July 1998.

[30] V. A. Bolotin. Modeling Call Holding Time Distributions for CCS Network Design and Performance Analysis. *IEEE Journal of Selected Areas In Communications*, 12(3):433–438, Apr. 1994.

[31] M. S. Borella. Source Models of Network Game Traffi c. *Computer Communications*, 23(4):403–410, Feb. 15, 2000.

[32] G. E. Box and G. M. Jenkins. *Time series analysis: forecasting and control*. McGraw-Hill, London, UK, 1970.

[33] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview, June 1994. RFC 1633.

[34] C. M. Braun and J. Giroux. Arcade Video Games: Proxemic, Cognitive and Content Analyses. *Journal of Leisure Research*, 21(2):92–105, 1989.

[35] A. J. Byer and D. Abrams. A comparison of the triangular and two-sample taste-test methods. *Food Technology*, 7(4):185–187, Apr. 1953.

[36] M. Bylund and F. Espinoza. Using Quake III Arena to Simulate Sensors and Actuators when Evaluating and Testing Mobile Services. In *Short Talk presented at the CHI 2001 Conference on Human factors in computing systems*, Seattle, WA, USA, Mar. 2001.

[37] E. Castronova. Virtual Worlds: A First-Hand Account of Market and Society on the Cyberian Frontier. Technical Report 618, Center for Economic Studies and Ifo Institute for Economic Research, California State University, Fullerton, CA, USA, Dec. 2001. [cited 19 June 2002; 08:20 BST] Available from Internet: <`http://papers.ssrn.com/sol3/papers.cfm?abstract_id=294828`>.

[38] CERT Coordination Center. Smurf IP Denial-of-Service Attacks, Jan. 1998. CERT Advisory CA-1998-01, [cited 26 Sept. 2002; 11:42 BST] Available from Internet: <`http://www.cert.org/advisories/CA-1998-01.html`>.

[39] L. Chappell and R. Spicer. Is your Network Doomed? *NetWare Connection*, Jan./Feb. 1996. [cited 26 Sept. 2002; 11:18 BST] Available from Internet: <`http://www.nwconnection.com/jan-feb.96/doomed/`>.

[40] C. Chatfi eld. *The Analysis of Time Series*. Chapman & Hall, London, UK, fi fth edition, 1996.

[41] D. Cheriton. The V distributed system. *Communications of the ACM*, 31(3):314–333, Mar. 1988.

[42] S. Cheshire. Latency and the Quest for Interactivity, Nov. 1996. White paper commissioned by Volpe Welty Asset Management, L.L.C., for the Synchronous Person-to-Person Interactive

Computing Environments Meeting, [cited 26 May 2002; 13:32 BST] Available from Internet: <http://www.stuartcheshire.org/papers/LatencyQuest.ps>.

[43] D. M. Chiu. Some Observations on Fairness of Bandwidth Sharing. In *Proceedings of the 5th IEEE Symposium on Computers and Communications (ISCC)*, pages 125–131, Antibes, France, July 2000.

[44] T. Chiueh. Distributed systems support for networked games. In *Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS-VI)*, pages 99–104, Cape Cod, MA, USA, May 1997.

[45] J. C. Chuang and M. A. Sirbu. Pricing Multicast Communication: A Cost-Based Approach. In *Proceedings of the 8th Internet Society Conference (INET)*, Geneva, Switzerland, July 1998.

[46] D. Cohen. Specifications for the Network Voice Protocol (NVP), Jan. 1976. RFC 741.

[47] C. Crawford. *The Art of Computer Game Design*. Osborne/McGraw-Hill, Berkeley, CA, USA, 1982. [cited 22 June 2002; 09:40 BST] Available from Internet: <http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html>.

[48] N. Croal. Making a Killing at Quake. *Newsweek*, page 104, Nov. 22, 1997.

[49] E. Cronin, B. Filstrup, and S. Jamin. Cheat-Proofing Dead Reckoned Multiplayer Games. In *Proceedings of the 2nd International Conference on Application and Development of Computer Games*, Hong Kong, Jan. 2003.

[50] E. Cronin, B. Filstrup, A. R. Kurc, and S. Jamin. An Efficient Synchronization Mechanism for Mirrored Game Architectures. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 67–73, Braunschweig, Germany, Apr. 2002.

[51] M. E. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, Dec. 1997.

[52] M. F. Daneshmand, R. R. Roy, and C. G. Savolaine. Framework and requirements of Quality of Service for multimedia applications. In *Proceedings of the 1997 IASTED International Conference on Intelligent Information Systems (IIS '97)*, pages 466–474, Grand Bahama Island, Bahamas, Dec. 1997.

[53] Datacomm Research Company. Winning Business Strategies for Mobile Games, July 2002. [cited 20 Sept. 2002; 15:50 BST] Available from Internet: <http://www.datacommresearch.com/competitiveedge/summary/wbsmg.asp>.

[54] DataMonitor. Online Games and Gambling, 1999-2004, Nov. 1999.

[55] DataMonitor. The Future of Wireless Gaming, Sept. 2000.

[56] R. Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, UK, 1976.

[57] G. Day. Online games: crafting persistent-state worlds. *IEEE Computer*, 34(10):111–112, Oct. 2001.

[58] K. E. Dill and J. C. Dill. Video game violence; A review of the empirical literature. *Aggression and Violent Behavior*, 3(4):407–428, Winter 1998.

[59] J. S. Duesenberry. *Income, Saving, and the Theory of Consumer Behavior*. Harvard University Press, Cambridge, MA, USA, 1949.

[60] D. E. Duffy, A. A. McIntosh, M. Rosenstein, and W. Willinger. Statistical Analysis of CCSN/SS7 Traffic Data from Working CCS Subnetworks. *IEEE Journal of Selected Areas In Communications*, 12(3):544–551, Apr. 1994.

[61] N. Economides and C. Himmelberg. Critical Mass and Network Evolution in Telecommunications. In G. W. Brock, editor, *Toward a Competitive Telecommunications Industry: Selected Papers from the 1994 Telecommunications Policy Research Conference*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1995.

[62] Inside Sony Online Entertainment. *Edge*, 102:56–61, Oct. 2001.

[63] H. J. Einsiedler and P. Hurley. Link Weighting: An Important Basis for Charging in the Internet. In *Proceedings of Global Internet '98*, Sydney, Australia, Nov. 1998.

[64] European Leisure Software Publishers Association (ELSPA): Weekly Chart Summary, May 05, 2002. [cited 18 June 2002; 09:10 BST] Available from Internet: <`http://www.elspa.com/research/chart.asp`>.

[65] J. F¨arber. Network Game Traffic Modelling. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 53–57, Braunschweig, Germany, Apr. 2002.

[66] J. R. Faria. Scientific, business and political networks in academia. *Research in Economics*, 56(2):187–198, June 2002.

[67] A. Feldmann, A. C. Gilbert, W. Willinger, and T. G. Kurtz. The Changing Nature of Network Traffic: Scaling Phenomena. *Computer Communication Review*, 28(2):5–29, Apr. 1998.

[68] S. Fischer, A. Hafid, G. von Bochmann, and H. de Meer. Cooperative QoS Management for Multimedia Applications. In *Proceedings of the 4th IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, pages 303–310, Ottawa, Canada, June 1997.

[69] F. Fitzek, G. Schulte, and M. Reisslein. System Architecture for Billing of Multi-Player Games in a Wireless Environment using GSM/UMTS and WLAN Services. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 58–64, Braunschweig, Germany, Apr. 2002.

[70] E. Frécon and M. Stenius. DIVE: A Scaleable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal*, 5(3):91–100, Sept. 1998.

[71] FreeCiv. [cited 29 May 2002; 10:44 BST] Available from Internet: <`http://www.freeciv.org`>.

[72] M. Fuchs and S. Eckermann. From "First-Person Shooter" to Multi-user Knowledge Spaces. In *Proceedings of Computational Semiotics for Games and New Media (COSIGN) 2001*, pages 83–87, Amsterdam, The Netherlands, Sept. 2001.

[73] T. Funkhouser. Network Topologies for Scalable Multi-User Virtual Environments. In *Proceedings of the Virtual Reality Annual International Symposium 1996 (VRAIS '96)*, pages 222–229, Santa Clara, CA, USA, Apr. 1996.

[74] S. Gallagher and S. H. Park. Innovation and competition in standard-based industries: A historical analysis of the US home video game market. *IEEE Transactions on Engineering Management*, 49(1):67–82, Feb. 2002.

[75] GameSpy. [cited 26 May 2002; 13:25 BST] Available from Internet: <`http://www.gamespy3d.com`>.

[76] C. M. Gorriz and C. Medina. Engaging Girls with Computers Through Software Games. *Communications of the ACM*, 43(1):42–49, Jan. 2000.

[77] R. Gossweiler, R. J. Laferriere, M. L. Keller, and R. Pausch. An Introductory Tutorial for Developing Multiuser Virtual Environments. *Presence: Teleoperators and Virtual Environments*, 3(4):255–264, Fall 1994.

[78] Graphics Visualization and Usability Center, Georgia Institute of Technology. 10th WWW User Survey, Dec. 1998. [cited 26 May 2002; 13:36 BST] Available from Internet: <`http://www.gvu.gatech.edu/user_surveys/survey-1998-10/`>.

[79] C. Greenhalgh and S. Benford. MASSIVE: a collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction*, 2(3):239–261, Sept. 1995.

[80] C. Greenhalgh, S. Benford, and M. Craven. Patterns of network and user activity in an inhabited television event. In *Proceedings of the ACM symposium on Virtual Reality Software and Technology*, pages 34–41, London, UK, Dec. 1999.

[81] M. Griffiths. Does Internet and Computer 'Addiction' Exist?: Some Case Study Evidence. In *Internet Research and Information for Social Scientists (IRISS 98)*, Bristol, UK, Mar. 1998. [cited 28 May 2002; 22:24 BST] Available from Internet: <`http://www.sosig.ac.uk/iriss/papers/paper47.htm`>.

[82] C. Griwodz. State replication for multiplayer games. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 29–35, Braunschweig, Germany, Apr. 2002.

[83] G. Hardin. The Tragedy of the Commons. *Science*, 162:1243–1248, Dec. 1968.

[84] K. Harrenstien, M. Stahl, and E. Feinler. NICNAME/WHOIS, Oct. 1985. RFC 954.

[85] T. Henderson. Latency and user behaviour on a multiplayer game server. In *Proceedings of the 3rd International Workshop on Networked Group Communication (NGC)*, pages 1–13, London, UK, Nov. 2001.

[86] T. Henderson. Observations on game server discovery mechanisms. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 47–52, Braunschweig, Germany, Apr. 2002.

[87] T. Henderson and S. Bhatti. Modelling user behaviour in networked games. In *Proceedings of the 9th ACM Multimedia Conference*, pages 212–220, Ottawa, Canada, Oct. 2001.

[88] T. N. Henderson and S. N. Bhatti. Protocol-independent multicast pricing. In *Proceedings of the 10th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 11–17, Chapel Hill, NC, USA, June 2000.

[89] D. Henriet and H. Moulin. Traffi c-based cost allocation in a network. *RAND Journal of Economics*, 27(2):332–345, Summer 1996.

[90] S. Herzog, S. Shenker, and D. Estrin. Sharing the "Cost" of Multicast Trees: An Axiomatic Analysis. In *Proceedings of ACM SIGCOMM '95*, pages 315–327, Cambridge, MA, USA, Aug. 1995.

[91] B. M. Hill. A simple general approach to inference about the tail of a distribution. *The Annals of Statistics*, 3(5):1163–1174, Sept. 1975.

[92] P. Hills and M. Argyle. Uses of the Internet and their relationships with individual differences in personality. *Computers in Human Behavior*, 19(1):59–70, Jan. 2003.

[93] H. Hotelling. Stability in Competition. *Economic Journal*, 39(153):41–57, Mar. 1929.

[94] F. hsiung Hsu. Computer Chess, Then And Now: The Deep Blue Saga. In *Proceedings of the 1997 International Symposium on VLSI Technology, Systems, and Applications*, pages 153–156, Taipei, Taiwan, June 1997.

[95] T. Ingvaldsen, E. Klovning, and M. Wilkins. Determining the causes of end-to-end delay in CSCW applications. *Computer Communications*, 23(3):219–232, Feb. 01, 2000.

[96] Institute of Electrical and Electronic Engineers. *1278.2-1995, IEEE Standard for Distributed Interactive Simulation — Communication Services and Profi les*. IEEE, New York, NY, USA, Apr. 1996.

[97] Institute of Electrical and Electronic Engineers. *1516-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) — Framework and Rules*. IEEE, New York, NY, USA, Sept. 2000.

[98] International Hobo <spiral@ihobo.com>. Re: [Digiplay] Etymology of "frag", Oct. 07, 2002. E-mail to digiplay mailing list <digiplay@topica.com>, [cited 12 Oct. 2002; 15:25 BST] Available from Internet: <`http://topica.com/lists/digiplay/read/message.html?mid=905457028`>.

[99] International Organization for Standardization. *ISO 690-2:1997: Information and documentation — Bibliographic references — Part 2: Electronic documents or parts thereof*. International Organization for Standardization, Geneva, Switzerland, Nov. 2001.

[100] International Telecommunication Union. *ITU-T Recommendation H.323: Packet-Based Multimedia Communications Systems*. International Telecommunication Union, Geneva, Switzerland, Feb. 1998.

[101] International Telecommunication Union. *ITU-R Recommendation BT.500: Methodology for the subjective assessment of the quality of television pictures*. International Telecommunication Union, Geneva, Switzerland, Mar. 2000.

[102] International Telecommunication Union. *ITU-T Recommendation G.114: International telephone connections and circuits — General Recommendations on the transmission quality for an entire international telephone connection — One-way transmission time*. International Telecommunication Union, Geneva, Switzerland, May 2000.

[103] Internet Software Consortium. Internet Domain Survey, July 2001. [cited 17 Sept. 2002; 17:07 BST] Available from Internet: <http://www.isc.org/ds/WWW-200107/dist-bynum.html>.

[104] ITU Internet Reports 2002. Internet for a Mobile Generation. Technical report, International Telecommunication Union, Geneva, Switzerland, Sept. 2002.

[105] J. Jacobson and Z. Hwang. Unreal Tournament for immersive interactive theater. *Communications of the ACM*, 45(1):39–42, Jan. 2002.

[106] V. Jacobson, C. Leres, and S. McCanne. tcpdump. [cited 20 Sept. 2002; 10:49 BST] Available from Internet: <http://www.tcpdump.org>.

[107] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB, June 1999. RFC 2598.

[108] B. E. John and A. H. Vera. A GOMS Analysis of a Graphic, Machine-Paced, Highly Interactive Task. In *Proceedings of the CHI '92 Conference on Human factors in computing systems*, pages 251–258, Monterey, CA, USA, May 1992.

[109] S. K. Joyce. Traffi c on the Internet —A study of Internet games, Oct. 2000. A report submitted in partial fulfi llment of the requirements for the degree of Bachelor of Computing and Mathematical Sciences, [cited 26 May 2002; 13:36 BST] Available from Internet: <http://wand.cs.waikato.ac.nz/wand/publications/sarah-420.ps.gz>.

[110] M. L. Katz and C. Shapiro. Network Externalities, Competition, and Compatibility. *American Economic Review*, 75(3):424–440, June 1985.

[111] F. P. Kelly. Charging and rate control for elastic traffi c. *European Transactions on Telecommunications*, 8(1):33–37, Jan./Feb. 1997.

[112] P. Key, L. Massoulié, and J. K. Shapiro. Service Differentiation for Delay-Sensitive Applications: An Optimisation-Based Approach. *Performance Evaluation*, 49(1-4):471–489, Sept. 2002.

[113] O. Kharif. Excuse Me, I've Got To Take This Game. *BusinessWeek Online*, July 02, 2001. [cited 06 Feb. 2003; 22:19 GMT] Available from Internet: <http://www.businessweek.com/bwdaily/dnflash/jul2001/nf2001072_760.htm>.

[114] I. Kouvelas, V. Hardman, and J. Crowcroft. Network Adaptive Continuous-Media Applications Through Self-Organised Transcoding. In *Proceedings of the 8th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 241–255, Cambridge, UK, July 1998.

[115] Lag City —The Quake Lag Page. [cited 13 June 2002; 12:45 BST] Available from Internet: <http://www.planetquake.com/lagcity/>.

[116] J. C. Laird. It knows what you're going to do: adding anticipation to a Quakebot. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 385–392, Montreal, Canada, May 2001.

[117] E. Lamb. A cross section of privately held game companies. *Red Herring*, 96:100–102, Apr. 15, 2001.

[118] J. Larkin. Winning the Monster Game. *Far Eastern Economic Review*, page 32, Sept. 05, 2002.

[119] J. Lawry, R. Upitis, M. Klawe, A. Anderson, K. Inkpen, M. Ndunda, D. Hsu, S. Leroux, and K. Sedighian. Exploring Common Conceptions About Boys and Electronic Games. *Journal of Computers in Math and Science Teaching*, 14(4):439–459, 1995.

[120] A. Legout, J. Nonnenmacher, and E. Biersack. Bandwidth Allocation Policies for Unicast and Multicast Flows. In *Proceedings of the 18th IEEE Conference on Computer Communications (INFOCOM)*, volume 1, pages 254–261, New York, NY, USA, Mar. 1999.

[121] H. Leibenstein. Bandwagon, Snob, and Veblen Effects in the Theory of Consumers' Demand. *Quarterly Journal of Economics*, 64(2):183–207, May 1950.

[122] W. E. Leland and D. V. Wilson. High time-resolution measurement and analysis of LAN traffic: Implications for LAN interconnection. In *Proceedings of the 10th IEEE Conference on Computer Communications (INFOCOM)*, volume 3, pages 1360–1366, Bal Harbour, FL, USA, Apr. 1991.

[123] T. Lenoir. All but War Is Simulation: The Military-Entertainment Complex. *Configurations*, 8(3):289–335, Fall 2000.

[124] E. Lety, L. Gautier, and C. Diot. MiMaze, a 3D Multi-player Game on the Internet. In *Proceedings of the 4th International Conference on Virtual Systems and Multimedia*, Gifu, Japan, Nov. 1998.

[125] S. Levy. *Hackers*. Penguin Books, London, UK, 1984.

[126] M. Lewis and J. Jacobson. Game Engines in Scientific Research. *Communications of the ACM*, 45(1):27–31, Jan. 2002.

[127] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:5–53, June 1932.

[128] G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, Aug. 1978.

[129] I. S. MacKenzie and C. Ware. Lag as a Determinant of Human Performance in Interactive Systems. In *Proceedings of the CHI '93 Conference on Human factors in computing systems*, pages 488–493, Amsterdam, The Netherlands, Apr. 1993.

[130] J. K. MacKie-Mason and H. R. Varian. Some Economics of the Internet. In W. Sichel and D. L. Alexander, editors, *Networks, Infrastructure and the New Task for Regulation*. University of Michigan Press, 1996.

[131] T. Manninen. Interaction in Networked Virtual Environments as Communicative Action —Social Theory and Multi-player Games. In *Proceedings of the Sixth International Workshop on Groupware (CRIWG2000)*, pages 154–157, Madeira, Portugal, Oct. 2000.

[132] T. Manninen. Virtual Team Interactions in Networked Multimedia Games —Case: "Counter-Strike" —Multi-player 3D Action Game. In *Proceedings of the 4th Annual International Workshop on Presence (PRESENCE 2001)*, Philadephia, PA, USA, May 2001.

[133] L. Mathy, C. Edwards, and D. Hutchison. Principles of QoS in group communications. *Telecommunication Systems*, 11(1-2):59–84, 1999.

[134] M. Mauve. How to keep a dead man from shooting. In *Proceedings of the 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS)*, pages 199–204, Enschede, The Netherlands, Oct. 2000.

[135] M. Mauve, S. Fischer, and J. Widmer. A Generic Proxy System for Networked Computer Games. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 25–28, Braunschweig, Germany, Apr. 2002.

[136] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proceedings of ACM SIGCOMM '96*, pages 117–130, Palo Alto, CA, USA, Aug. 1996.

[137] J. F. McCarthy. Active Environments: Sensing and Responding to Groups of People. *Journal of Personal and Ubiquitous Computing*, 5(1):75–77, 2001.

[138] S. McCreary and K. Claffy. Trends in Wide Area IP Traffi c Patterns: A View from Ames Internet Exchange. In *Proceedings of the ITC Specialist Seminar on IP Traffi c Modeling, Measurement and Management*, Monterey, CA, USA, Sept. 2000. [cited 14 June 2002; 12:51 BST] Available from Internet: <`http://www.caida.org/outreach/papers/2000/AIX0005/`>.

[139] A. I. McLeod. Diagnostic Checking Periodic Autoregression Models with Application. *The Journal of Time Series Analysis*, 15(2):221–233, 1994.

[140] E. Medina. Yahoo enters game rental arena. *Boston Globe*, page C3, Sept. 23, 2002.

[141] R. M. Metcalfe. The Internet After the Fad. In *The 1996 Monticello Lectures*, Monticello, VA, USA, May 1996. [cited 26 May 2002; 13:37 BST] Available from Internet: <`http://www.americanhistory.si.edu/csr/comphist/montic/metcalfe.htm`>.

[142] Microsoft Xbox. [cited 3 July 2002; 16:01 BST] Available from Internet: <`http://www.microsoft.com/xbox`>.

[143] S. Miller. On the effects of network packet loss on Half-Life. Technical report, University College London, London, UK, 2002. Project submitted as part requirement for the M.Sc in Computer Science.

[144] J. Milnor. A Nobel Prize for John Nash. *The Mathematical Intelligencer*, 17(3):11–17, Summer 1995.

[145] D. Min, E. Choi, D. Lee, and B. Park. A load balancing algorithm for a distributed multimedia game server architecture. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 882–886, Florence, Italy, June 1999.

[146] M. Minges. Counting the Net: Internet Access Indicators. In *Proceedings of the 10th Internet Society Conference (INET)*, Yokohama, Japan, July 2000.

[147] J. Morahan-Martin and P. Schumacher. Incidence and correlates of pathological Internet use among college students. *Computers in Human Behavior*, 16(1):13–29, Jan. 2000.

[148] netfi lter/iptables project. [cited 24 June 2002; 10:19 BST] Available from Internet: <`http://netfilter.samba.org`>.

[149] NetValue. Internet Overview —Your Key To the Internet Landscape, Feb. 2001. [cited 29 May 2002; 20:14 BST] Available from Internet: <`http://www.netvalue.com/corp/pdf/internet_overview_promosheet.pdf`>.

[150] N. H. Nie and L. Erbring. Internet and Society —A Preliminary Report. Technical report, Stanford Institute for the Quantitative Study of Society, Stanford, CA, USA, Feb. 16, 2000. [cited 26

May 2002; 13:38 BST] Available from Internet: <http://www.stanford.edu/group/siqss/Press_Release/Preliminary_Report.pdf>.

[151] NIST Net network emulation package. [cited 18 Sept. 2002; 16:51 BST] Available from Internet: <http://snad.ncsl.nist.gov/itg/nistnet/>.

[152] Nokia N-Gage. [cited 06 Feb. 2003; 21:23 GMT] Available from Internet: <http://www.n-gage.com>.

[153] P. Olla and N. V. Patel. A value chain model for mobile data service providers. *Telecommunications Policy*, 26(9-10):551–571, Oct.-Nov. 2002.

[154] V. N. Padmanabhan and L. Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *Proceedings of ACM SIGCOMM 2001*, pages 173–185, San Diego, CA, USA, Aug. 2001.

[155] W. H. Page and J. E. Lopatka. Network Externalities. In B. Bouckaert and G. D. Geest, editors, *Encyclopedia of Law and Economics*, pages 952–980. Edward Elgar, Aldershot, UK, 2000.

[156] 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 System Support for Digital Audio and Video (NOSSDAV)*, pages 23–29, Miami Beach, FL, USA, May 2002.

[157] L. Pantel and L. C. Wolf. On the Suitability of Dead Reckoning Schemes for Games. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 79–84, Braunschweig, Germany, Apr. 2002.

[158] K. S. Park and R. V. Kenyon. Effects of network characteristics on human performance in a collaborative virtual environment. In *Proceedings of IEEE VR 1999*, pages 104–111, Houston, TX, USA, Mar. 1999.

[159] A. Patrizio. Coming Soon: Pay-Per-Game. *Wired News*, Oct. 20, 2000. [cited 26 May 2002; 13:25 BST] Available from Internet: <http://www.wired.com/news/culture/0,1284,39505,00.html>.

[160] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[161] L. Phlips. *The Economics of Price Discrimination*. Cambridge University Press, Cambridge, UK, 1983.

[162] A. Postlewaite. The social basis of interdependent preferences. *European Economic Review*, 42(3-5):779–800, May 1998.

[163] J. A. Price. Social Science Research on Video Games. *Journal of Popular Culture*, 18(4):111–125, Spring 1985.

[164] J. M. Pullen and D. C. Wood. Networking Technology and DIS. *Proceedings of the IEEE*, 83(8):1156–1167, Aug. 1995.

[165] QStat. [cited 26 May 2002; 13:26 BST] Available from Internet: <http://www.qstat.org>.

[166] H. Rachlin. Altruism and Selfishness. *Behavioral and Brain Sciences*, Aug. 2001. In press, [cited 21 Oct. 2002; 11:33 BST] Available from Internet: <http://www.bbsonline.org/Preprints/Rachlin/>.

[167] M. Ranta-aho, A. Leppinen, G. Poulain, A. Roella, M. Mirabelli, A. Ousland, and J. Norgaard. Task-dependent user requirements for Quality of service of Videoconferencing-CSCW services. In *Proceedings of the 16th International Symposium on Human Factors in Telecommunications*, pages 251–254, Oslo, Norway, May 1997.

[168] K. Ravindran and T.-J. Gong. Cost Analysis of Multicast Transport Architectures in Multiservice Networks. *IEEE/ACM Transactions on Networking*, 6(1):94–109, Feb. 1998.

[169] D. P. Reed. Going Nowhere Fast. *Context Magazine*, July/Aug. 1999. [cited 18 June 2002; 17:10 BST] Available from Internet: <`http://www.contextmag.com/archives/199907/TheGreatLie.asp`>.

[170] D. P. Reed. Weapon of Math Destruction. *Context Magazine*, Spring 1999. [cited 26 May 2002; 13:39 BST] Available from Internet: <`http://www.contextmag.com/archives/199903/DigitalStrategy.asp`>.

[171] Resource ranges allocated by APNIC. [cited 31 May 2002; 16:07 BST] Available from Internet: <`http://www.apnic.net/db/ranges.html`>.

[172] L. Rizzo. Dummynet: A Simple Approach to the Evaluation of Network Protocols. *Computer Communication Review*, 27(1):31–41, Jan. 1997.

[173] L. Rizzo, L. Vicisano, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proceedings of the 17th IEEE Conference on Computer Communications (INFOCOM)*, volume 3, pages 996–1003, San Francisco, CA, USA, Mar. 1998.

[174] F. Rose. Pocket Monster. *Wired*, 9(9), Sept. 2001.

[175] N. Schactman. EverQuest: The Latest Addiction. *Wired News*, July 29, 1999. [cited 19 June 2002; 07:55 BST] Available from Internet: <`http://www.wired.com/news/culture/0,1284,20984,00.html`>.

[176] N. Schactman. The Real Second Coming of Diablo. *Wired News*, May 22, 2000. [cited 28 May 2002; 15:49 BST] Available from Internet: <`http://www.wired.com/news/technology/0,1284,36263,00.html`>.

[177] C. Schaefer, T. Enderes, H. Ritter, and M. Zitterbart. Subjective Quality Assessment for Multiplayer Real-Time Games. In *Proceedings of the First Workshop on Network and System Support for Games (NetGames2002)*, pages 74–78, Braunschweig, Germany, Apr. 2002.

[178] K. Schmidt and L. Bannon. Taking CSCW Seriously: Supporting Articulation Work. *Computer Supported Cooperative Work: An International Journal*, 1(1-2):7–40, 1992.

[179] Sega Dreamcast. [cited 3 July 2002; 15:50 BST] Available from Internet: <`http://www.sega.com/games/dreamcast/hardware.jhtml`>.

[180] B. Shelley. Transcript of presentation on Aesthetics of Game Design. In *Computer and Video Games come of age: A national conference to explore the state of an emerging entertainment medium*. MIT Program in Comparative Media Studies, Feb. 2000. [cited 22 June 2002; 11:47 BST] Available from Internet: <`http://web.mit.edu/cms/games/aesthetics.html`>.

[181] S. R. Sherman. Perils of the Princess: Gender and Genre in Video Games. *Western Folklore*, 56(3 and 4):243–258, Summer/Fall 1997.

[182] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press, Boca Raton, FL, USA, 1997.

[183] Z. B. Simpson. The In-game Economics of Ultima Online. In *Proceedings of the 14th Games Developers Conference*, San Jose, CA, USA, Mar. 2000. [cited 19 June 2002; 08:24 BST] Available from Internet: <http://www.gdconf.com/archives/proceedings/2000/simpson.doc>.

[184] S. Singhal and M. Zyda. *Networked Virtual Environments — Design and Implementation*. ACM Press, New York, NY, USA, 1999.

[185] Sky Digital. [cited 22 Oct. 2002, 11:02 BST] Available from Internet: <http://www.sky.com/skycom/article/0,,70043-1046815,00.html>.

[186] Slashdot.org. How Fast Too Slow? A Study Of Quake Pings, May 24, 2001. [cited 28 May 2002; 10:33 BST] Available from Internet: <http://slashdot.org/articles/01/05/24/2044233.shtml>.

[187] A. Smith. *The Theory of the Moral Sentiments*. A. Millar, London, UK, 1759.

[188] Sony PlayStation 2. [cited 3 July 2002; 16:02 BST] Available from Internet: <http://uk.playstation.com/hardware/ps2Console.jhtml>.

[189] Sophos Anti-Virus plc. Sophos virus analysis: W32/Rodok-A. [cited 11 Oct. 2002, 18:18 BST] Available from Internet: <http://www.sophos.com/virusinfo/analyses/w32rodoka.html>.

[190] H. Stone, J. Sidel, S. Oliver, A. Woolsey, and R. C. Singleton. Sensory Evaluation by Quantitative Descriptive Analysis. *Food Technology*, 28(11):24–34, Nov. 1974.

[191] R. J. Swickert, J. B. Hittner, J. L. Harris, and J. A. Herring. Relationships among Internet use, personality, and social support. *Computers in Human Behavior*, 18(4):437–451, July 2002.

[192] M. Terrano and P. Bettner. 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond. In *Proceedings of the 15th Games Developers Conference*, San Jose, CA, USA, Mar. 2001. [cited 26 Sept. 2002; 14:59 BST] Available from Internet: <http://www.gdconf.com/archives/proceedings/2001/terrano_1500arch.doc>.

[193] The JNT Association. The JANET Report 2000-2001, July 2001. [cited 25 Oct. 2002; 20:55 BST] Available from Internet: <http://www.ja.net/documents/janetreport/report2001.pdf>.

[194] F. D. Tran, M. Deslaugiers, A. Gérodolle, L. Hazard, and N. Rivierre. An open middleware for large-scale networked virtual environments. In *Proceedings of the IEEE Virtual Reality Conference 2002*, pages 22–29, Orlando, FL, USA, Mar. 2002.

[195] Ultima Online —Advanced Character Service. [cited 19 Sept. 2002; 07:06 BST] Available from Internet: <http://support.uo.com/advancedcharacter.html>.

[196] Urban Mercenary. [cited 26 May 2002; 13:26 BST] Available from Internet: <http://www.urbanmercenary.com>.

[197] I. Vaghi, C. Greenhalgh, and S. Benford. Coping with inconsistency due to network delays in collaborative virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 42–49, London, UK, Dec. 1999.

[198] J. L. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, USA, 1944.

[199] A. Watson and M. A. Sasse. Measuring Perceived Quality of Speech and Video in Multimedia Conferencing Applications. In *Proceedings of the 6th ACM Multimedia Conference*, pages 55–60, Bristol, UK, Sept. 1998.

[200] J. Widmer, M. Mauve, and J. P. Damm. Probabilistic Congestion Control for Non-Adaptable Flows. In *Proceedings of the 12th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 13–21, Miami Beach, FL, USA, May 2002.

[201] D. Williams. Structure and Competition in the U.S. Home Video Game Industry. *The International Journal on Media Management*, 4(1):41–54, 2002.

[202] R. B. Williams and C. A. Clippinger. Aggression, competition and computer games: computer and human opponents. *Computers in Human Behavior*, 18(5):495–506, Sept. 2002.

[203] S. Williamson, M. Kosters, D. Blacka, J. Singh, and K. Zeilstra. Referral Whois (RWhois) Protocol V1.5, June 1997. RFC 2167.

[204] W. Willinger, V. Paxson, and M. S. Taqqu. Self-similarity and Heavy Tails: Structural Modeling of Network Traffi c. In R. J. Adler, R. E. Feldman, and M. S. Taqqu, editors, *A Practical Guide to Heavy Tails — Statistical Techniques and Applications*, pages 27–53. Birkh¨auser, Boston, MA, USA, 1998.

[205] D. J. Zizzo and A. J. Oswald. Are People Willing to Pay to Reduce Others' Incomes? *Annales d'Economie et de Statistique*, 63-64:39–66, July/Dec. 2001.