Networked games — a QoS-sensitive application for QoS-insensitive users?

Tristan Henderson[*]
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
T.Henderson@cs.ucl.ac.uk

Saleem Bhatti
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
S.Bhatti@cs.ucl.ac.uk

ABSTRACT

Research into providing different levels of network Quality of Service (QoS) often assumes that there is a large market for QoS-sensitive applications that will be fulfilled once QoS-enabled networks have been deployed. Multiplayer networked games are an example of such an application that requires QoS, and hence will only become popular if QoS is made widely available. The *prima facie* evidence, however, is that games are already popular, in spite of the existing QoS-free best-effort Internet.

Networked games may have become popular despite the lack of QoS because players "make do" with what is available to them. Such popularity is a double-edged sword. It may mean that there is a demand, as yet unfulfilled, from game players for QoS-enabled networks. On the other hand, it may mean that players have become accustomed to playing games without QoS, and therefore might be less willing to pay for higher QoS when it does eventually become available.

In this paper we present the results of a short experiment to examine the QoS tolerances of game players. We use a set of popular First Person Shooter (FPS) game servers that are publicly available to Internet users at large. By systematically altering the network latency to the servers, we attempt to study whether degraded QoS (in the form of higher network delay) affects a user's decision to participate in the game.

We find that increased network delay has an effect on a user's decision to join a game server. It appears, however, that there is no significant difference in the number of players who leave the game as a result of increased delay. We speculate that this may be due to a user's enjoyment exceeding their QoS-sensitivity, and discuss the implications of our findings with respect to providing and charging for QoS.

[*]Tristan Henderson was sponsored by a Hewlett-Packard EPSRC CASE award.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ACM SIGCOMM 2003 Workshops August 25 & 27, 2003, Karlsruhe, Germany
Copyright 2003 ACM 1-58113-748-6/03/0008S ...$5.00.

1. INTRODUCTION

For many years, researchers have investigated and developed methods for providing different levels of Quality of Service (QoS) in data networks. Perhaps the most famous examples are the IETF standards for Integrated Services (IntServ) [5] and Differentiated Services (DiffServ) [4]. Comparatively little effort, however, has been expended on determining which applications may benefit from networks that offer a menu of different levels of QoS. Instead, it is assumed that there is a variety of QoS-sensitive applications, which can be categorised into classes such as "Real Time" and "Non-Real Time", or by tasks, such as "Multimedia Collaboration" or "Video-on-Demand" [7]. Once QoS-aware networks are deployed, the market for these QoS-sensitive applications will become satisfied, thereby justifying the expense of enabling QoS in the network.

Multiplayer networked games are one example of a real-time QoS-sensitive application. They are perhaps one of the most interesting examples, because unlike other real-time applications such as multimedia conferencing, games have already become popular amongst Internet users. For instance, the Sony MMORPG (Massively Multiplayer Online Role-Playing Game) *Everquest* features 400,000 regular users, each playing for an average of 20 hours a week [8], whilst the Korean MMORPG *Lineage* claims that four million users play on its servers [17]. Measurement studies also indicate that games are responsible for an increasingly large amount of network traffic [20].

The popularity of multiplayer networked games is puzzling, since the current Internet can only provide a best-effort service. The levels of delay, loss and throughput that human factors research indicates to be a requirement for multimedia applications, cannot be guaranteed in the existing Internet. In spite of this, hundreds of thousands of users of the best-effort Internet are happy to use this network to play fast-paced real-time networked games.

One possible explanation for the popularity of networked games on the Internet is that players are simply "making do" with best-effort service, since they lack the option to acquire higher levels of QoS. They would rather play games over a lossy and high-delay network, than not play games at all.

The flipside to having millions of players already making do with best-effort service, however, is that if and when QoS-enabled networks do become deployed, the higher levels of QoS that such networks make available will have to offer significant benefits over best-effort service. Price-sensitive players may not see the point in paying for levels of QoS that only offer incremental improvements over the existing networked games.

Whether price-sensitive players will be willing to pay for QoS may depend in some part on whether they are able to detect the effects of higher or lower levels of QoS. A player who cannot tell the difference between a best-effort and a priority service will be unlikely to pay for the latter. In this paper, we examine whether current game players can detect different levels of QoS. We concentrate on one aspect of network QoS, namely, network latency or delay. By altering the delay between game players and a public game server, we test whether users are dissuaded by degraded QoS.

This paper is organised as follows. In Section 2 we outline the QoS requirements for networked games and discuss why games have been categorised as a QoS-sensitive application. Section 3 outlines the experiments described in this paper and the methodology used. In Section 4 we describe our results, and in Section 5 we discuss the implications of these results and some ideas for future work.

2. QOS REQUIREMENTS FOR GAMES

Mathy *et al.* [19] list the five QoS parameters that 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

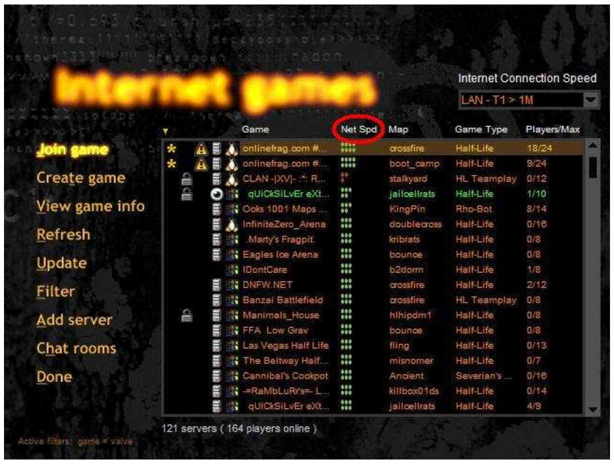
Different applications will have different requirements for each of these parameters — for instance, a video conference might require low jitter, but tolerate a high level of loss, whereas a shared whiteboard might require no loss, but tolerate low bandwidth.

Throughput is typically not an issue for current networked games, and most games are designed to operate at a worst case over dial-up links. Instead, several researchers assert that delay is the most important parameter of performance for a networked multimedia application [29, 30]. In particular, network delay is judged to be a large problem for game players. On the popular online discussion forum Slashdot [33], one can find comments 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 therefore appear that some game players believe that the delay bounds for networked games to be tight, and a requirement for such an application.

The importance of low delay in networked games is well-known to game designers, who have devised methods to attempt to conceal or repair the effects of high network latency [3]. The issue of delay is also made apparent to end-users in some games. Figure 1 shows the in-game server browser for the game *Half-Life*. The circled column marked "Net Spd" is intended to provide an indication of the network delay between the player and a given game server, and thus the end-user is able to choose between different game servers on the basis of their round-trip time.



Internet Connection Speed: LAN - T1 > 1M

Game	Net Spd	Map	Game Type	Players/Max
onlinefrag.com #...		crossfire	Half-Life	18/24
onlinefrag.com #...		boot_camp	Half-Life	9/24
CLAN -[XV]- :*: R...		stalkyard	HL Teamplay	0/12
qUiCkSiLvEr eXt...		jailcellrats	Half-Life	1/10
Ooks 1001 Maps ...		KingPin	Rho-Bot	8/14
InfiniteZero_Arena		doublecross	Half-Life	0/16
.Marty's Fragpit.		kribrats	Half-Life	0/8
Eagles Ice Arena		bounce	Half-Life	0/8
IDontCare		b2dorm	Half-Life	1/8
DNFW.NET		crossfire	Half-Life	2/12
Banzai Battlefield		crossfire	HL Teamplay	0/8
Manimals_House		hlhipdm1	Half-Life	0/8
FFA Low Grav		bounce	Half-Life	0/8
Las Vegas Half Life		fling	Half-Life	0/13
The Beltway Half...		misnomer	Half-Life	0/7
Cannibal's Cookpot		Ancient	Severian's ...	0/16
-=RaMbLuRr's=- L...		killbox01ds	Half-Life	0/14
qUiCkSiLvEr eXt...		jailcellrats	Half-Life	4/9

121 servers (164 players online)

Active filters: game = valve

Figure 1: *Half-Life* game server browser

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 [2]. The IEEE Distributed Interactive Simulation (DIS) standard stipulates a latency bound of between 100ms and 300ms for military simulations [13]. MacKenzie and Ware find that in a VR (Virtual Reality) environment, interaction becomes very difficult above a delay of 225ms [18]. 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 [14]. Park and Kenyon [27] examine a two-user cooperative task in an Networked Virtual Environment (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 [25] 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.* [31] 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.* [35] analyse the effects of delay in a simple ball game implemented on the MASSIVE NVE , and 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 [6] proposes a latency bound of 100ms for networked games, although no empirical basis is given for this.

The most popular networked game genres are currently the aforementioned MMORPG, the RTS (Real-Time Strategy) and the FPS (First Person Shooter). Of these games, the FPS game is perhaps the most delay-sensitive, as it entails users running around shooting at each other in real-time, whereas the MMORPG and RTS games involve user interaction at a slightly slower pace. Armitage studies the FPS game *Quake* and finds that network latencies above 150ms appear to dissuade game players [1]. In previous work, we have looked at a similar FPS game, *Half-Life*, and finds that players tend not to play when latencies are above 225-250ms [11].

3. METHODOLOGY

We have been running public game servers for the FPS game *Half-Life* for some time. There are several advantages to using an FPS game over an MMORPG or RTS game for experiments. MMORPGs tend to have a small number of servers with a large number of participants. These servers are controlled by companies who typically do not allow non-commercial access. In comparison, FPS games have a large number of servers, with a small number of participants per server. The software to run a *Half-Life* server is included with the game, and there are anywhere between 2,500 to 20,000 servers running on the Internet at any given time [12]. It is therefore easy to set up servers for research purposes. The servers that we have been running are registered under a non-academic URL, and are only advertised through the standard mechanisms built in to the game. They should thus appear identical to any of the other servers on the Internet from an end-user's perspective.

For the experiments described in this paper, we used a pair of identical *Half-Life* game servers, 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. We refer to these servers in this paper as `server1` and `server2` respectively. We configured the servers so as to disable any delay-concealing techniques [3]. Disabling this "lag compensation" at the server overrides any client-side settings; it does not, however, disable delay-concealing mechanisms that are purely client-side, such as dead reckoning. The servers were 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 [22] were used to introduce delay into the network. An *iptables* filter was set up on the gateway to queue packets that were addressed to the IP address and port number of the game server. These packets were queued in userspace and placed back on the network queue after the desired period of time had passed.

The servers were left to run for two months to build up a regular userbase. Once this had occurred, we added additional delay to one of the servers, so as to degrade the QoS to that server. This additional delay alternated between the two servers, so that users would not begin to ignore one of the servers.

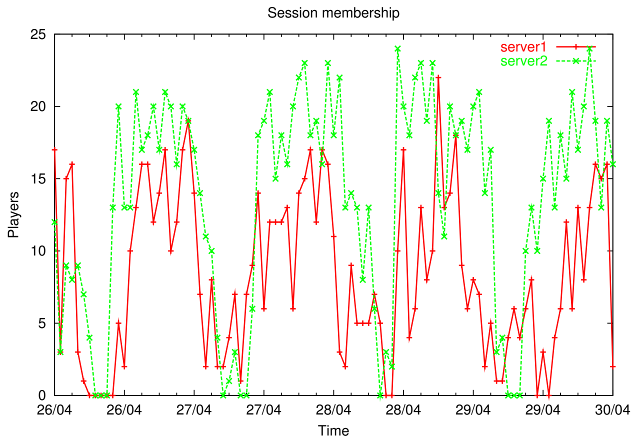
We hypothesised that higher levels of delay could affect users in two ways:

1. They would be dissuaded from joining the server at all. This might be by looking at the higher delay in server browsers (Figure 1), or by joining the server, observing the high latency and disconnecting.
2. Players already on the server might leave the server after observing their higher latency.

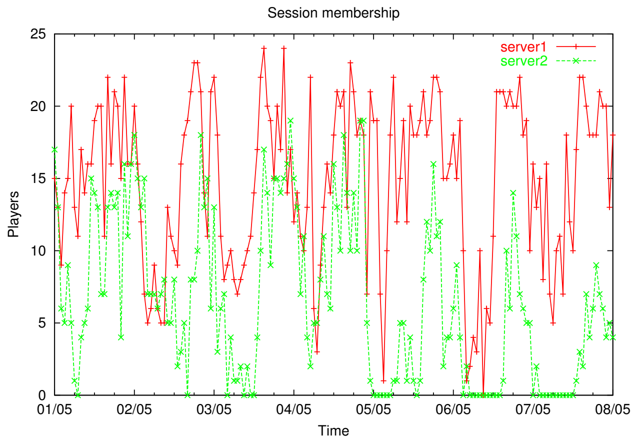
We thus designed our experiments to test these two different possibilities.

4. RESULTS

4.1 Joining a server



(a) Additional delay to `server1`



(b) Additional delay to `server2`

Figure 2: Players on two servers with differing levels of network delay

To test whether higher levels of delay affect a user's decision to join a game server, 50ms of delay was added to one of the servers, and the number of players on both servers was monitored. Figures 2 and 3 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 2 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, `server1` (the solid line) had additional delay, and so there are fewer players on that server (Figure 2(a)). From 02/05/02 to 08/05/02, `server2` (the dashed line) had additional delay, and the situation reverses (Figure 2(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 3 shows a similar time period to Figure 2, except that there was no additional delay added to either server. 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 therefore conclude that network delay does have an effect on a user's decision to join a game server.

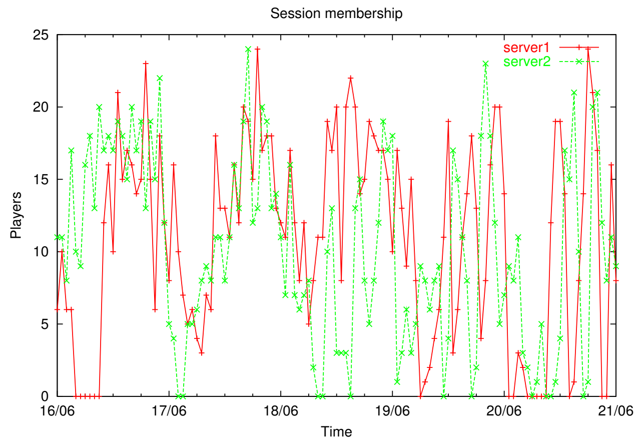


Figure 3: Players on two servers with no additional network delay

4.2 Leaving a server

To examine the effects of delay on a user's decision to leave a server, another 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 of the players observed on the server, 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 4 plots the percentage of players on the server that chose 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). Each of the six boxplots in this Figure comprises a "box" that indicates the first to third quantile of the observations at each level of delay, while the horizontal line in the box indicates the median, and the "whiskers" indicate the extreme values. By examining the median values, it can be seen that as the level of additional delay increases, there is little change in the percentage of players that chose to leave the server, which remains 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 the metric *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.

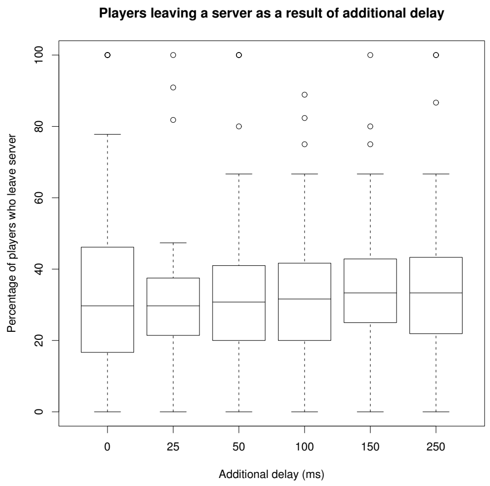


Figure 4: Players leaving a server as a result of additional delay

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 highly engrossed in the game's virtual world. We examined the duration of a player's session 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. It appears that the longer that 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.

Another factor that might affect a user's decision to leave the server is the number of times that they had already played on the server. There might be hysteresis effects for regular players; they might be used to playing on a particular server, or have friends on the server, and thus be more willing to put up with degraded QoS to stay on their chosen server. We found, however, that 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), $t(2895) = 1.12, p = 0.2628$.

Players might choose to remain on the server in spite of the additional delay because 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.

4.3 Leaving a server due to differences in relative delay

By introducing delay to all the players on the server, 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, i.e., their relative delay was higher than the other players, 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 4.2 was repeated, but instead of introducing additional delay to all the players, 20% of the players were randomly chosen to receive additional delay.

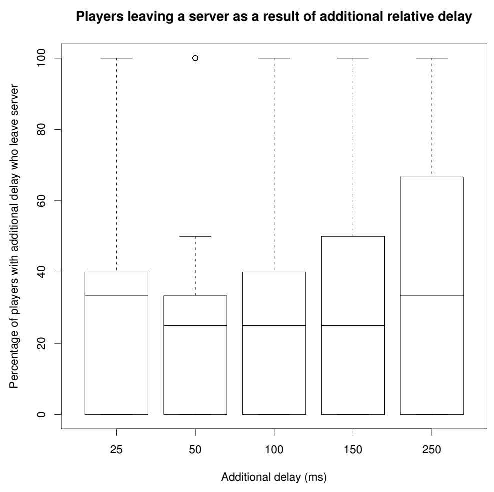


Figure 5: Players leaving a server as a result of additional relative delay

Figure 5 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 4). 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.

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, $t(3937) = 17.6115, p < 0.01$.

Adding delay to some of the players on the server created effects that 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. DISCUSSION AND FUTURE WORK

These experiments indicate that degraded QoS can be noticed by networked game players, in that it can dissuade them from joining a game server. Once game players are connected to a server, however, it appears that degraded QoS in the form of higher network delay does not cause them to leave the server. Moreover, players who have been on the server for a longer period of time are less likely to leave in the event of higher delay.

Our experiments have been very short, and there are parts of our methodology that may require further refinement. As with all experiments that take place over the public Internet, there are many variables that have been out of our control. We do not know whether players left or chose to stay on the server for exogenous reasons. For instance, they may have been able to detect the higher delay, but ignored it, believing that it was due to transient congestion, or because they wanted to carry on with the game regardless. We believe, however, that the advantages of using public servers, such as the larger number of participants, outweigh the drawbacks.

As the experiments presented here have been brief, there are several improvements that can be addressed in future work. It might be interesting to only add delay to the same players each time, rather than a random selection, or targeting those players with higher skill, to see if they can better adapt to higher delay. Other possible modifications might be to alter the length and periodicity of the additional delay, or to increase the level of additional delay.

In this paper we have only examined one facet of network QoS, network delay. Although it is commonly assumed that delay is the most important QoS factor for networked games, it would be useful to further examine game players' tolerances for different levels of network throughput, jitter and loss. Mobile gaming has often been cited as a "killer application" for the expensive third-generation wireless phone networks, and such networks are susceptible to much higher levels of jitter and loss than the fixed networks on which most online gaming currently takes place. It would also be interesting to examine different types of games, as studies of MMORPG, RTS and driving games [34, 25, 32] have indicated that their delay requirements differ from those of the FPS game.

Our results might be skewed by the nature of the application that we have examined. The *Half-Life* game offers players the chance to choose between different game servers on the basis of their delay, as we discussed in Section 2, and it has been demonstrated that players do tend to choose servers with lower delay [1]. Thus players may select servers to which they have low latency at the time of joining. Once in the game, however, a player has to actively choose to monitor their delay, by bringing up the "Scoreboard" which lists the number of "frags" (kills) and the network delay for each player connected to the server. We have conducted surveys of game players that indicate that many players do not check their delay on a regular basis during the game [24]. One reason why players seem to be more tolerant of delay once they have connected to a server, might be that they are no longer checking their delay, and so are unaware of any added latency. On the other hand, the fact that increased delay is not an issue for the players unless they are made aware of it (i.e., by the act of checking the scoreboard) implies that perhaps delay is not as big a hindrance to gameplay as is commonly assumed.

The above limitations and caveats aside, the results in this paper indicate that players are willing to tolerate higher levels of delay than the human factors literature indicates. They are also willing to tolerate higher levels of delay than our own previous work has shown [11], and in some cases a delay increase of 100% was not enough to dissuade players. We have stated that we believe that players are more delay-sensitive when choosing a server than once they are connected and playing the game. Thus, it is unsurprising that the player tolerances we present here appear to be higher than those on an unadulterated game server.

If users are not leaving a game server because they are unable to detect the degraded QoS, then it means that they would be unlikely to pay for higher, but undetectable, levels of QoS. We have found that many game players are unwilling to pay for higher levels of QoS [24]. Many of the game players that we have interviewed would prefer the cost of better network performance to be included in the price of the game, or to be transparent to the user. This preference for inclusive charges is reflected in the more successful game business models — companies that have attempted to charge "per-play" [28] have been unable to gain the large audiences of games, such as *Everquest* and *Lineage*, that charge all-inclusive monthly fees. This predilection by end-users for "flat rate" pricing has also been noted for Internet access and other services [9, 23]. Yahoo!, however, has recently launched an online rental games portal [21], and it remains to be seen whether this will be a failure or success.

If game players cannot detect higher levels of delay, or can detect but choose to ignore them, then this has interesting implications for providing QoS for networked games. It might not be optimal for a network provider to offer QoS policies such as Assured [10] or Expedited Forwarding [15], where QoS preferences are predetermined for the life of a flow or application. Instead, a policy where higher levels of QoS are offered at the beginning of a player's session, to entice them into the game, and slowly degrading QoS as the players become more engrossed in the game's virtual world, might be preferable from a network provider's point of view. Of course, this is suboptimal from the player's point of view, especially if they were paying for a given level of QoS. Players might thus choose to alter their behaviour if offered such a QoS policy, for instance by disconnecting and reconnecting so as to receive the initially-higher QoS. Several researchers have considered non-cooperative network users in regard to provisioning QoS [16, 26]. The game-theoretic issues of a non-cooperative ISP who wishes to deceive their users by providing dynamically-degrading QoS policies are a potential avenue for future work.

6. REFERENCES

[1] G. Armitage. Sensitivity of Quake3 players to network latency. Poster, *ACM SIGCOMM Internet Measurement Workshop 2001*, Berkeley, CA, USA, Nov. 2001.

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

[3] 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.

[4] S. Blake, D. L. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services, Dec. 1998. RFC 2475.

[5] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet architecture: an overview, June 1994. RFC 1633.

[6] 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.

[7] 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.

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

[9] P. C. Fishburn and A. M. Odlyzko. Competitive pricing of information goods: Subscription pricing versus pay-per-use. *Economic Theory*, 13(2):447–470, Mar. 1999.

[10] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group, June 1999. RFC 2597.

[11] 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.

[12] T. Henderson. Observations on game server discovery mechanisms. In *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames)*, pages 47–52, Braunschweig, Germany, Apr. 2002.

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

[14] 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.

[15] V. Jacobson, K. Nichols, and K. Poduri. An expedited forwarding PHB, June 1999. RFC 2598.

[16] Y. A. Korilis, A. A. Lazar, and A. Orda. Architecting noncooperative networks. *IEEE Journal of Selected Areas In Communications*, 13(7):1241–1251, Sept. 1995.

[17] J. Larkin. Winning the monster game. *Far Eastern Economic Review*, page 32, Sept. 05, 2002.

[18] 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.

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

[20] S. McCreary and K. Claffy. Trends in wide area IP traffic patterns: A view from Ames Internet Exchange. In *Proceedings of the ITC Specialist Seminar on IP Traffic Modeling, Measurement and Management*, Monterey, CA, USA, Sept. 2000.

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

[22] netfilter/iptables project. `http://netfilter.samba.org`.

[23] A. M. Odlyzko. Internet pricing and the history of communications. *Computer Networks*, 36(5-6):493–517, Aug. 2001.

[24] M. Oliveira and T. Henderson. What online gamers really think of the Internet. In *Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames)*, pages 177–185, Redwood City, CA, USA, May 2003.

[25] 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.

[26] K. Park, M. Sitharam, and S. Chen. Quality of service provision in noncooperative networks: heterogenous preferences, multi-dimensional QoS vectors, and burstiness. In *Proceedings of the 1st International Conference on Information and Computation Economies (ICE-98)*, pages 111–127, Charleston, SC, USA, Oct. 1998.

[27] K. S. Park and R. V. Kenyon. Effects of network characteristics on human performance in a collaborative virtual environment. In *Proceedings of the IEEE Virtual Reality Conference (VR '99)*, pages 104–111, Houston, TX, USA, Mar. 1999.

[28] A. Patrizio. Coming soon: Pay-per-game. *Wired News*, Oct. 20, 2000. `http://www.wired.com/news/culture/0,1284,39505,00.html`.

[29] 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.

[30] D. P. Reed. Going nowhere fast. *Context Magazine*, July/Aug. 1999.

[31] C. Schaefer, T. Enderes, H. Ritter, and M. Zitterbart. Subjective quality assessment for multiplayer real-time games. In *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames)*, pages 74–78, Braunschweig, Germany, Apr. 2002.

[32] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The effect of latency on user performance in Warcraft III. In *Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames)*, pages 3–14, Redwood City, CA, USA, May 2003.

[33] Slashdot.org. How fast too slow? A study of Quake pings, May 24, 2001. `http://slashdot.org/articles/01/05/24/2044233.shtml`.

[34] 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.

[35] 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.