Latency and User Behaviour on a Multiplayer Game Server

Tristan Henderson*

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

Abstract. Multiplayer online games represent one of the most popular forms of networked group communication on the Internet today. We have been running a server for a first-person shooter game, Half-Life. In this paper we analyse some of the delay characteristics of different players on the server and present some interim results. We find that whilst network delay has some effect on players' behaviour, this is outweighed by application-level or exogenous effects. Players seem to be remarkably tolerant of network conditions, and absolute delay bounds appear to be less important than the relative delay between players.

1 Introduction

Multiplayer online games represent one of the most popular forms of networked group communication on the Internet today, and they contribute to an increasingly large proportion of network traffic [11]. There has been little work to analyse or characterise these applications, or to determine any specific user or network requirements. The real-time nature of many of these games means that response times are important, and in a networked environment this means that round-trip delays must be kept to a minimum. Is network delay, however, the most important factor in a user's gaming experience? In this paper we examine the relationship between application-level delay and player behaviour in multiplayer networked games. The main question that we wished to answer was "How important is delay in a player's decision to select and stay on a particular games server?". To achieve this, we have been running a publicly-accessible server for one of the more popular FPS (First Person Shooter) games, Half-Life [13]. From this server, we have logged and analysed usage behaviour at both the application and network level. The paper is structured as follows. In Section 2 we look at previous work and discuss some expectations we had prior to this study. Section 3 describes our server setup and data collection methodology. Section 4 describes the results that we observed, and Section 5 outlines directions for further work.

* The author is funded by a Hewlett-Packard EPSRC CASE award.

2 Background

In this section we discuss previous work on multiplayer games, and what this led us to expect before commencing this study.

Previous analysis of popular commercial networked games has thus far concentrated on observing local area network traffic and behaviour [2, 4], and network-level rather than user- and -session-level characteristics. There is also little empirical analysis of the delay requirements for real-time multiplayer applications. However, it is generally accepted that low latencies are a requirement. Cheshire [5] proposes 100ms as a suitable bound, whilst human factors research indicates that 200ms might be a more appropriate limit [1]. The IEEE DIS (Distributed Interactive Simulation) standard stipulates a latency bound of between 100ms and 300ms for military simulations [8]. MacKenzie and Ware find that in a VR (Virtual Reality) environment, interaction becomes very difficult above a delay of 225ms [10]. Such previous work implies that there should be an absolute bound to player delay, beyond which players' gameplay becomes so impaired that they would either abort the game or find another server.

MiMaze was a multiplayer game which ran over the multicast backbone (MBone). Some statistics related to a network session of the game are presented in [6]. Using a sample of 25 players, they find that the average client delay is 55ms, and as such the state synchronisation mechanisms are designed with a maximum delay of 100ms in mind. The limited nature of the MBone, however, means that the measurements taken might not be representative of games today. The average delay is far lower than what we observe, and might result from the fact that all clients were located within the same geographic region (France) and well-connected to each other via the MBone.

Like MiMaze, Half-Life uses a client-server architecture, but the distribution model is unicast. Players connect to a common server, which maintains state about the nature of the game. The objective of the game is to shoot and kill as many of the other players as possible. This setup is representative of most of the popular FPS games. Games are typically small, between 16 and 32 players, and there are several thousand servers located all over the Internet. Table 1 shows the average number of servers for some of the more popular games. We obtained these figures by querying master servers for these games every 12 hours for two months. Since there are lots of small groups on servers located all over the Internet, it is reasonable to assume that players will attempt to connect to one of the servers with the lowest delay. This implies that most players would come from geographic locations near to our server, assuming that these have lower delays.

The nature of FPS games means that delay should be an important factor in the gaming experience. Players run around a large "map" (the virtual world) picking up weapons and firing at each other. Low response times should therefore be an advantage, since players can then respond more successfully to other users. If, however, the main benefit of low delay is to gain an advantage over other players, then we might expect that absolute delay is not so important as the

Game	*Average number of servers*
Half-Life	15290.52
Unreal Tournament	2930.16
Quake III Arena	2217.93
Quake II	1207.43
Tribes 2	968.67
QuakeWorld	389.28
Quake	84.94
Sin	49.05
Tribes	42.51
Kingpin	42.20
Heretic II	9.12

Table 1. Average number of servers for different FPS games

variance of delay, if a player needs only to have a relatively low delay compared to the other players in the game.

Usability analysis of non-networked computer games, e.g. [9] indicates that many actions become routine for players as they become expert at the tasks involved, but whether this still holds when the opponents are less predictable (i.e., other humans) is unclear. Unfortunately, we know of no usability studies that specifically analyse networked games, but perhaps regular and highly-skilled players might be able to better tolerate delay, as they become more adept at the specific details and strategy of a particular game, such as Half-Life.

3 Methodology

We recorded users than connected to a games server that we set up at University College London (UCL) in the UK. This server comprised a 900MHz AMD Athlon PC with 256 Mb of RAM, running Linux kernel version 2.4.2, and was connected to our departmental network via 100BaseT Ethernet. To prevent the possibility of users being prejudiced by connecting to an academic site, we registered a non-geographic .com domain name to use instead of a cs.ucl.ac.uk address. 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.

The game was set to rotate maps 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 only 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.

Player behaviour was monitored at both the application and the network level. For application-level logging, we took advantage of the server daemon's built-in logging facilities, and augmented this with an additional third-party server management tool to provide more comprehensive logs. Packet-level monitoring used tcpdump, which was set to log UDP packet headers only.

The data that is analysed here derives from running the server between 21 March 2001 18:33 GMT and 15 April 2001 08:28 BST. In this time we observed 31941 sessions (a single user joining and leaving the server).

3.1 Determining unique users

Many of the issues that we examine in Section 4 require knowledege of which sessions correspond to which particular users, for example, examining the average delay across all of a particular players' sessions. Such persistent user/session relationships cannot be determined by network-level traces alone, and session-level data is required. However, the nature of most FPS games, where any user can connect to any 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`). 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 player purchases the Half-Life software. There is thus one unique WONID for each purchased copy of the game. 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 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 or the use of pirate copies of the game, and so the same WONID can 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 493 times during the period of this study). In addition, on two occasions the WON Authentication Server seemed to malfunction, issuing 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 we needed to try and determine which sessions belonged to which different individuals.

The identifying information logged by the server for each player is the player's WONID, their IP address and port number, and the nickname that they choose

to use in the game. Of the 14776 total WONIDs that we observed, 11612 had unique (WONID, nickname, IP address, port) tuples; we can be reasonably sure that each of these represents one unique user. Of the remaining WONIDs, we had to make some assumptions about users in order to determine their uniqueness. 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. Users tend to change their names quite often, 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. When multiple users with the same WONID were simultaneously connected we assume that these represent different players. Using these heuristics, we estimate that the 14776 WONIDs actually represent 16969 users.

3.2 Measuring delay

To measure the delay observed by each player we used the game server's built-in facilities. The server was set up to log the application-level round-trip delay every 30 seconds. Of the total 1314007 measurements, we removed the 16306 with a value of 0, assuming that they are errors. We also saw 10592 measurements greater than 1000ms, with a maximum of 119043ms. We remove these measurements, also assuming that they are errors, since it is unlikely that any user would be able to play a networked game effectively with a 2 minute delay. Moreover, a similar FPS game, Quake III Arena, also assumes that client delays over 1000ms are errors, but chooses not to report them to users, and so we do the same here.

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. In any case, 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. Informal testing showed us that on a lightly-loaded client, the delays reported by Half-Life are within 5ms of a network-level ping, but, unsurprisingly, this difference rises with client and server load. Unfortunately, without access to the source code of the game, we cannot be sure what causes the erroneous ($\geq$ 1000ms) delays.

Although Half-Life does include features for refusing players admission depending on their delay, and for compensating for variations in player delays [3], these were disabled on our test server, since these might influence any results concerning relative delays.

In addition to measuring the application-level delay, we also performed a whois lookup on players' IP addresses in order to obtain some indication of their area of origin. Further details of the methodology and results of this analysis,

and anonymised versions of the server logs, can be found at the author's webpage (`http://www.cs.ucl.ac.uk/staff/T.Henderson`).

4 Results

Our main results can be summarised as follows:

- There is a wide distribution in players' average delay, with over 40% of players experiencing delays of over 225ms.
- Delay does not appear to play a part in a player's decision to return to the server, or to stay on the server.
- Most players connect during 1800-2400, according to their respective time-zones.
- There is some correlation between a player's ability and their delay and session duration.
- Social bonds do not appear to have an effect on player behaviour.

4.1 Absolute delay

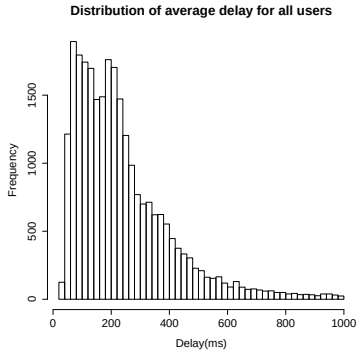
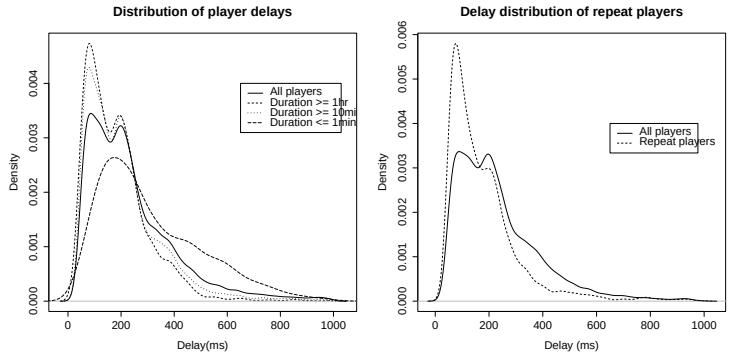


Fig. 1. Distribution of players' average delay

Figure 1 shows the distribution of the average delay observed over the duration of each player's session. The largest proportion of players appear to have delays of between 50 and 300ms, while 95% of players have delays under 533ms (Table 2). The large number of users with high delays of over 225ms is interesting since gameplay should theoretically be quite difficult at this level. However, Figure 1 also includes the delays of "tourists"; those players who connect to a server, examine the status of the game and then choose to leave. Figure 2(a) shows the distribution of delay for all the players compared to those who stay less than a minute, and those who stay more than 10 minutes and 1 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. A player with a delay of over 400ms is 2.68 times as likely to stay for less than one minute. This implies that delay is a determinant in a player's decision to join a server; players with high delays to a particular server will look elsewhere.

If, however, 100-225ms represents the upper delay bounds for interaction, then we would expect that most of the players with delay above this level would choose other servers. Yet 40.56% of the players who stay for more than one minute, have average delays of over 225ms, and there is no significant difference in the duration of players with delays over 225ms.



(a) Distribution of players' average delay

(b) Distribution of repeat players' delay

Fig. 2. Kernel density functions of delay distribution

We define regular players as those who played more than 10 times and whose average session duration exceeded one minute. There were 279 such players. Figure 2(b) indicates that the repeat players' mean delay tends to be lower, but this is statistically insignificant at a 5% confidence level.

Players	*Mean delay (ms)*	*95th percentile*
All	232	533
Regular	176	424
Tourists	339	733

Table 2. Overall delay results

4.2 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.

We measure the relative delay in two ways. First, we look at the nominal results and calculate the standard deviation of players' delay to give an estimate of range. Secondly, we analyse the ordinal data and look at a player's rank in terms of delay compared to the other players.

For most of the time, there is a deviation of around 100ms between players (Figure 3). This seems reasonable, given that most players' delay is in the region of 100-200ms.

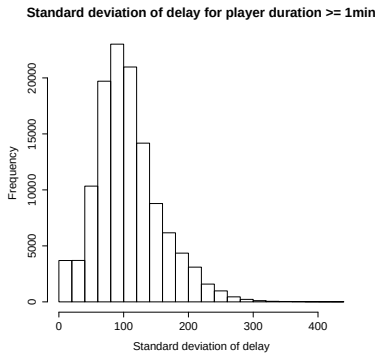
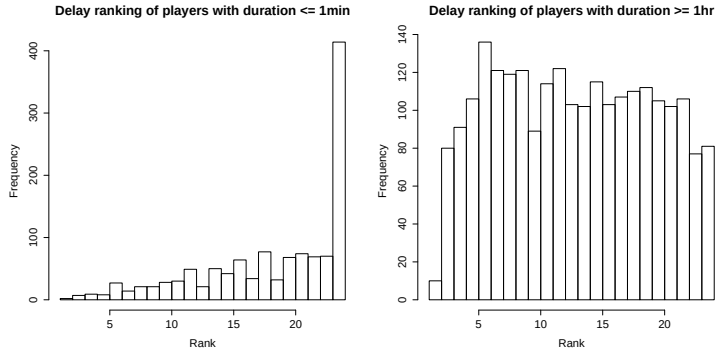


Fig. 3. Distribution of standard deviation of delay

The "delay rank" of a particular player was calculated by ordering the players at each delay measurement period by delay to produce a rank r, and then scaling the number of players n against the potential maximum number of players of 24, i.e. $r \times 24/n$. Thus, the player with the highest delay would always receive a delay rank of 24, whereas the minimum possible score of 1 would only be possible if the player had the lowest delay and there were 24 players on the server. Figure 4 indicates some correlation between the delay ranks; players who leave tend to have a higher rank.

4.3 Leaving a game

If delay is a determinant of user behaviour, then one might expect to see a change in delay towards the end of a user's session. A sudden increase in delay might lead a user to leave the server and connect to another one, or give up playing



(a) Delay ranks where player duration $\leq$ 1min

(b) Delay ranks where player duration $\geq$ 1hr

Fig. 4. Relative delay ranks

altogether. We see little evidence for this hypothesis, however. Figure 5(a) shows the "exit delay" (the delay over the last 5% of a player's session) compared to the average delay over the length of the session. This ratio congregates around the value of 1; i.e., the exit delay is usually comparable to the average delay.

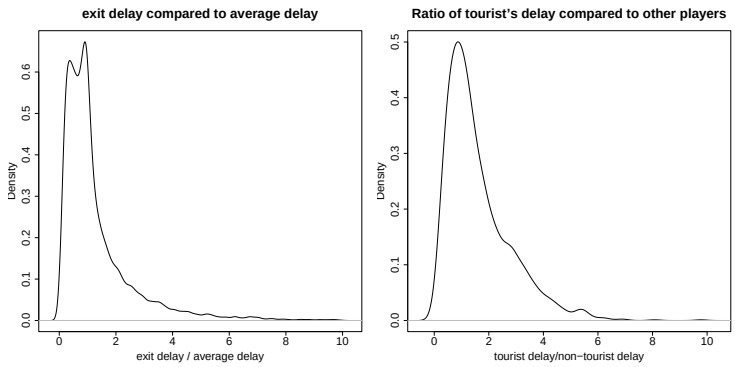
Relative delay does not seem to be a determinant of tourists leaving the server, either. Figure 5(b) shows the ratio of the delay of those tourists who join and leave the server, compared to the delay of the players already on the server. The mean is 1.618, and there is no correlation between the two.

4.4 When do players play?

Figure 6(a) shows the average number of players for each day of the week, sampled every thirty minutes. There is a strong time-of-day component, which agrees with our previously-observed results [7]. This is perhaps surprising given that players come from areas with different time zones; Figure 6(b) shows the number of players from Europe and North America (determined from the whois database). The offset in the respective peak times is probably due to time zones. Unsurprisingly, the peak usage times are in the late evening, from around 1800 to 2400. If users tend to play in their spare time, then perhaps they have already allocated this time for gameplay, and so are willing to put up with whatever network conditions they happen to experience.

4.5 Player skill

A player's ability might have an effect on the delay that they can tolerate. A user who is highly skilled at playing the game might be able to cope with higher delays



(a) Players' exit delay compared to average delay

(b) Tourist delay compared to other players

Fig. 5. Exit and tourists' delay

than beginners, since they might be able to predict other player's behaviour and thus compensate for higher-than-average lag. In human factors terms, a high level of skill might lead to players being able to perform actions without conscious awareness — in other words, playing the game becomes automatic.

The session-level logs include details of which players killed each other, and with which particular weapon. Using this information it is possible to estimate of the skill of each player. Whenever a kill takes place, we calculate the players' skill using the following formula:

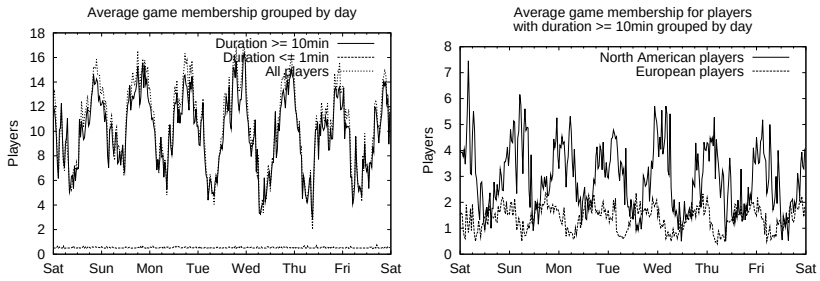
$$s_k = s_k + s_k/s_d * w; s_d = s_d - s_k/s_d * w$$

where s_k = killer's skill, s_d = killed player's skill, and w is an adjustment for the weapon used (e.g., it is harder to kill with a crowbar than a machine gun).

Using this metric, we see no correlation between skill and delay, nor skill and the duration of a player's game. We see some positive correlation ($R = 0.378$) between session duration and skill, so the more expert players do tend to stay longer. There is a slight negative correlation ($R = -0.231$) between skill and delay, so a lower delay may lead to improved performance.

4.6 Social bonds

Figure 2(b) indicates the presence of certain players who had excessively high delays, yet kept returning to the server. Here we analyse some of these players in more detail to see if it is possible to determine why they keep coming back. Table 3 shows some of these players' statistics.



(a) Average number of players each day

(b) Average number of North American and European players each day with duration $\geq$ 10 min

Fig. 6. Number of players over time

TLD (from whois)	*Number of visits*	*Average delay (ms)*	*Maximum delay (ms)*	*Average duration (sec)*	*Maximum duration (sec)*
TW	10	423	533	390	1460
TH	38	794	953	2319	10418
KR	14	340	389	2477	10043
TW	15	554	771	665	1659
KR	13	339	362	970	2553

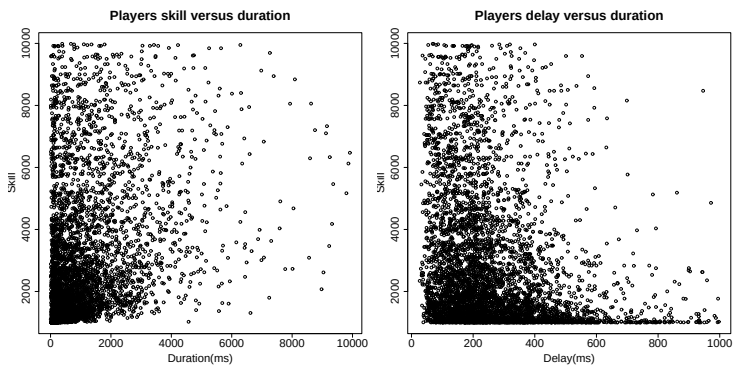
Table 3. Detailed statistics for regular players with high delays

One possibility is that these players are returning to the server because of friends who are also playing. However, of the 283 other players who were on the server at the same time as these five players, only 33 players appear twice and one appears three times. It is therefore unlikely that repeat visits or social bonds were the reason for these players returning.

5 Conclusions, caveats and future work

This study has looked at the effects of delay on user dynamics on a multiplayer games server. We find that application-level delay does not appear to have a significant effect on a user's behaviour once they have chosen to connect to a games server. Although the majority of users have delays within the bounds predicted by previous VR and DIS studies, changes in this delay do not seem to lead to players aborting the game.

Although brief and still at an interim stage, this study has raised a number of interesting research questions. If users are concerned with relative delay, is



(a) Skill versus duration (b) Skill versus delay

Fig. 7. Effect of skill

it possible to design efficient algorithms for determining the server with the smallest standard deviation in delay given a group of prospective users? There is currently a lot of interest in optimal placement of web and mirror servers e.g. [12] and perhaps this could be extended to locating game servers.

That games players might be unaffected by sudden changes in delay has important implications for designing potential congestion control schemes for games. In particular, pricing schemes that depend on users adapting to network conditions because of price changes might be less practical. If users tend to remain in a game for an exogenously determined duration, then session-based pricing or reservations might make more sense from a user's perspective. Pricing schemes for games could be designed to adapt the network to the user (who has already committed to playing a game), rather than the other way around.

Usability studies of multiplayer networked FPS games, e.g. a GOMS (Goals, Operators, Methods and Selection) analysis such as that performed in [9], might help to explain some of the results we have seen, since simple correlations of kills and deaths appear to be insufficient. More elaborate skill metrics, e.g. taking into account the amount of time between deaths, might also prove fruitful.

As our results are only from the study of a single Half-Life server, which may not be representative of servers across the Internet as a whole, we intend to investigate lightweight methods for instrumenting larger numbers of servers for future data collection. In spite of these limitations, this study has provided us with some direction for future experimental work. This study has only been correlational — we have examined and attempted to interpret results from an unmodified server. In future work we intend to run multiple servers, modifying variables such as network delay and jitter, and simulating different congestion

control and QoS policies, to further investigate their effects on user behaviour. We expect to find our study corroborated by this further study.

Acknowledgement

Thanks to Saleem Bhatti, Jon Crowcroft and the reviewers for their comments.

References

1. R. W. Bailey. *Human Performance Engineering — Using Human Factors/Ergonomics to Achieve Computer System Usability.* Prentice Hall, Englewood Cliffs, NJ, second edition, 1989.
2. 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.
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, Mar. 2001.
4. M. S. Borella. Source models of network game traffic. *Computer Communications*, 23(4):403–410, Feb. 15, 2000.
5. 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.
6. L. Gautier and C. Diot. Design and evaluation of MiMaze, a multi-player game on the Internet. In *Proceedings of the 1998 IEEE International Conference on Multimedia Computing and Systems*, pages 233–236, Austin, TX, June 1998.
7. 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.
8. Institute of Electrical and Electronic Engineers. *1278.2-1995, IEEE Standard for Distributed Interactive Simulation — Communication Services and Profiles.* IEEE, New York, NY, Apr. 1996.
9. 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, May 1992.
10. 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.
11. 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, Sept. 2000.
12. L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *Proceedings of the 20th IEEE Conference on Computer Communications (INFOCOM)*, pages 1587–1596, Anchorage, AK, Apr. 2001.
13. Valve Software. Half-Life. `http://www.sierrastudios.com/games/half-life/`.