

Game Server Selection for Multiple Players

Steven Gargolinski, Christopher St. Pierre, and Mark Claypool
CS Department, Worcester Polytechnic Institute
100 Institute Road, Worcester, MA, 01609, USA
claypool@cs.wpi.edu

ABSTRACT

The increase in power and connectivity of computers has enabled a growth in network games, with many games having numerous servers to which a player can connect. The game server selected influences the game play, both by impacting the game type and map choice as well as the connection latency and server performance. Often, geographically spread-out friends, family and clan members want to play together on a centrally-located game server with good performance. Unfortunately, current game server selection tools only consider the perspective of a single client, with multiple players left to coordinate their game server selection manually through an out-of-band means, such as telephone or online chat. This manual server coordination process is time-consuming, at best, and error-prone, at worst, often resulting in the selection of a poorly performing game server. This paper presents an architecture that allows geographically dispersed players that want to play together to select the best game server for their shared game play. Implementation details of a working system based on the architecture are presented, including a preliminary evaluation illustrating the system's effectiveness.

Categories and Subject Descriptors: C.2.m [Computer-Communication Networks]: Miscellaneous

General Terms: Measurement, Performance

Keywords: Server selection, Network games

1. INTRODUCTION

The online component of video games has grown considerably over the past decade with some recent games being released with only online multi-player gameplay. Multiple player network computer games can make up around half of the top 25 types of non-traditional traffic for some Internet links [5] and are predicted to make up over 25% of Local Area Network (LAN) traffic by the year 2010.

Many network games allow players to choose which server

to connect to for their online play. For many games, this arises because individual users can run their own game servers, allowing clients to connect to their server from anywhere on the Internet. Nearly all popular first person shooters (such as Quake, Doom, and Battlefield) allow individual users to run game servers. Similarly, most real-time strategy games (such as Warcraft and Age of Mythology) allow users to host a game, thus providing many choices for clients playing online.

And the choice of game server matters. Game servers can become full, limiting a player's ability to join the server. The requirement of some game servers for clients to have cheat protection enabled (such as PunkBuster), or specific client versions installed may also physically limit a player's choices. The choice of the game map, game configuration and other in-game parameters (such as having friendly-fire disabled for a team-based first-person shooter) can determine a player's desire to join a particular game server.

Even if all physical and preferential game conditions are met by a game server, the network and server performance will impact the choice of the best server. The range of latencies from a client to all game servers can be as broad as the range of end-to-end Internet latencies. Moreover, game players care about application to application latencies, not just end-host to end-host latencies, so latency from server load adds to the network latency. Several studies have demonstrated the negative effects latency can have on player performance [1, 2, 6, 7], making selection of a close and fast server important for good online game play.

The problem of game server selection can be compounded when geographically separated players want to play together on the same game server. This can arise when friends and family hook up for a regular on-line gaming night or when more formalized teams of players (clans) compete against other teams on a regular basis. A server that is fast for player A may be slow for player B and vice versa. Finding a game server that performs well for all players that want to play together requires server information to be shared among the clients.

With current game server selection browsers, when a player starts a game client, the client contacts a master server that lists all game servers that are up at that time. The client then individually pings each server to get latency information as well as server configuration parameters (map in play, number of players, etc.). A player can then sort the resulting list of game server information in a variety of ways, such as by increasing latency or by server map type.

While somewhat flexible for one player, current game server

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.

NetGames'05, October 10–11, 2005, Hawthorne, New York, USA.
Copyright 2005 ACM 1-59593-157-0/05/0010 ...\$5.00.

browsers provides no support at all for selection of a good server that suits multiple players that want to play together. Instead, the players must manually collate information from their individual server browsers in order to find an appropriate game server. Using a communication mechanism external to the game (such as instant messaging or Internet phone), players typically offer and counter-offer game server choices one-by-one until a seemingly suitable choice is found. Not only is this tedious, but it is prone to sub-optimal solutions. A popular game on a busy night can provide thousands of server choices, making consideration of all servers impractical. Moreover, manually keeping track of performance parameters (such as the lowest overall latency or the fairest latency for all players) is prone to errors, especially when more than two players want to play together.

This paper proposes an architecture that allows geographically dispersed players that want to play together to select the best server for their game play. The idea is that optimal game server selection for multiple players requires that the game clients share information about their view of possible game servers with other game clients. In our approach, game clients gather server information in the same manner as they would for single player game play. Then, the server information for all clients is sent to a host that combines all the game client information, selecting the best server based on a customizable server selection algorithm. The game clients are then informed of the game server choice and each player joins in a game at the selected server. Preliminary evaluation illustrates the difficulties in selecting a game server for multiple players using state of the art (manual) means, and shows the promise of our approach.

The rest of this paper is organized as follows: Section 2 describes the state of the art in game server selection; Section 3 details our approach for server selection for multiple players; Section 4 presents a preliminary evaluation of the approach; Section 5 summarizes the paper; and Section 6 presents possible future work.

2. STATE OF THE ART

2.1 Game Server Selection for One Player

For many online games, players must specify the game server to which they want to connect before starting. A player chooses a game server via a server browser, a tool that obtains and displays game server information. The server browser contacts a well-known master server¹ to obtain a list of the active game servers, then sends individual ping packets to each active game server. The ping packets provide an estimate of the latency to each game server and contain information about the game type and game server. The server browser then allows the player to sort by various fields (number of players, game type, ping time, etc.). After selecting an appropriate game server, a player can launch the game, connecting to that server.

2.2 Game Server Selection for Multiple Players

Currently, the only way for two geographically separated players that want to play together to find a game server which performs well from both locations is through a te-

¹A list of master servers for many popular games can be found at <http://www.qstat.org/qstatdoc.html>.

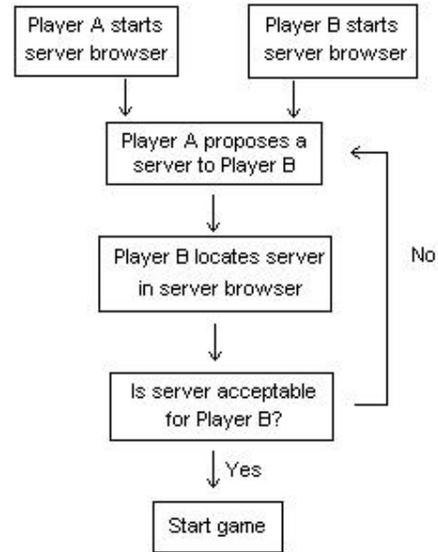


Figure 1: Manual Game Server Selection Flowchart for Two Players

dious, sub-optimal, trial-and-error method (which also requires some sort of external communication mechanism, such as a telephone or instant messaging).

As an example, assume Player A is connecting from San Francisco, California and wants to play with Player B, connecting from Tokyo, Japan. Figure 1 depicts a flowchart of the steps each player would need to carry out before a (hopefully) appropriate game server was found.

Players A and B start their server browser and obtain game information as if selecting a game server for a single player.

Player A begins with the servers that have the lowest ping times and proceeds through the list to the servers with the highest ping times. Player A suggests a possible game server to Player B. The server with the lowest ping time to San Francisco most likely does not have the lowest ping time to Tokyo. Therefore, Player B is required to scan through her list of servers until she is able to locate the specified server.

When both players have located the same server in their server browser, the next step is to look at the two ping values and decide if this server is acceptable. Assuming that the server in question is located in San Diego, it may well provide a low ping time from San Francisco, but will likely provide a significantly higher ping time across the Pacific Ocean from Tokyo and prove unacceptable.

There are then two basic options on how to proceed. In the first option, player A could move down his server list and propose a new game server to Player B, most likely the second best server on his list. At this point, the process discussed above would repeat itself with this new game server. Player B would find the new game server on her list, evaluate it, and make a decision. In the second option, Player B could make a counter-offer, finding a server at the top of her list and reporting it to Player A. Player A would locate the server in his own server browser, check the ping value, and decide if the server is acceptable for game play or not.

This process will repeat itself until a game server is proposed by Player A and accepted as playable by Player B (or vice-versa).

This method is both time-consuming and often yields sub-optimal results. For many games, the number of game servers available at any given time makes checking every possible server impractical, a problem that will be exacerbated in the future as online, multi-player games proliferate. Even evaluating a very a small percentage of these servers is extremely time-consuming. Assuming it takes 10 seconds to evaluate the effectiveness of one server, searching through the top-10 servers for each player takes over 3 minutes. This is often larger than the round-time in first person shooter games, meaning the game server conditions will likely have changed since the server selection process started. Because of this delay (not to mention any errors in doing the manual correlation), players end up making a game server choice which is probably not optimal.

The situation discussed assumes that only two players are interested in playing together. As we add more users into this scenario, the problem becomes even more tedious, time-consuming and error-prone.

2.3 QStat

QStat is a command-line program that gathers real-time statistics from Internet game servers [4], information that a game server browser can use to allow a game client to select an appropriate server. Most games that *QStat* supports are first-person shooters (Quake, Half-Life, etc.) and since most first-person shooters support server browsing using either the Unreal or Quake protocols, *QStat* also works with most first-person shooters.

QStat obtains the latency information for each game server by sending an application-level ping packet over UDP to the game server. Upon receiving the ping packet, the game server responds with basic game information. The data in the ping response can be formatted in several ways, including XML. Upon receiving the ping response, *QStat* records the latency as the time between sending the ping and receiving the response. In the event the ping or the ping response are lost (or the server is down), *QStat* will repeat the ping request two additional times.

For some examples, the following command obtains detailed server information for all Quake III servers registered with the master server:

```
qstat -q3m,68 master3.idsoftware.com -R -P
```

and the following command obtains basic server information from a specific Quake III server at port 27960:

```
qstat -q3s 216.12.96.41:27960
```

3. SERVER SELECTION FOR MULTIPLE PLAYERS

This section presents an architecture that supports game server selection for multiple players. The architecture features a master-slave system, where a central master process controls the slave processes run locally by game players on their clients. The master instructs the slaves to gather server information, transmit the information to the master and, ultimately, run the game on the server selected. A system implementation using the architecture is available for download at <http://www.cs.wpi.edu/~claypool/papers/musst/>.

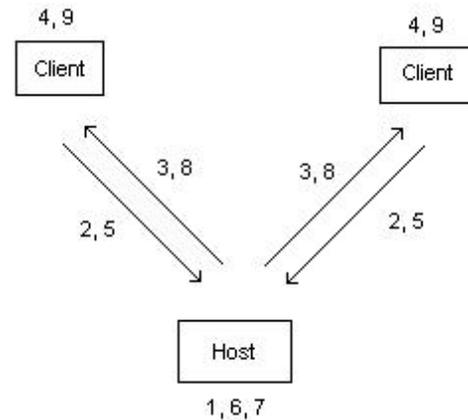


Figure 2: Architecture and Interaction for Game Server Selection for Multiple Players

3.1 Architecture

Typically, one player agrees to be a *host*, responsible for setting up the game. The other players act as *clients*, responsible only for connecting to the host. There is always exactly one host, who may or may not also be a client interested in playing in the game. Any number of players may run clients and connect to the host. The interaction between the host and the clients is depicted in Figure 2.

The host first selects game options of interest to the players, such as acceptable maps, game type (Deathmatch, Capture the Flag, etc.) and the number of players in the group that want to play together. The host then goes into a listening state, waiting for the specified number of clients to connect (step 1 in Figure 2).

Each client is then responsible for connecting to the host (step 2). To do this, clients need only to specify the IP address of the host. No other options are specified at the client side.

The host remains in the listening state until the number of players that want to play together have connected. At this point, the host uses the options specified in order to build a *QStat* command (see Section 2.3) which, when run, allows the clients to gather game server data from the game's master server. The server then transmits this command line data to each of the clients (step 3).

Upon receiving the command line options sent by the host, each client executes *QStat* and collects the *QStat* data returned, describing all relevant game servers in XML format (step 4). When the client finishes gathering the data, it transfers the data information back to the host (step 5).

The host receives XML data from the clients until it has collected a complete set of data from each connected client. The host parses the data files from each client, reading the information into internal data structures for further selection (step 6).

This data is then run through a series of algorithms (described in Section 3.2) to determine the best server for the players (step 7). The server selected is sent back to each client along with the proper command required to launch the chosen game and connect to the indicated game server

Player	Location	IP	ISP Connection
A	Hoorn, The Netherlands	82.217.15.247	Cable modem
B	Massachusetts, United States	130.215.239.33	T3
C	Kota Bharu, Malaysia	60.48.62.140	DSL
D	North Carolina, United States	67.77.3.251	DSL

Table 1: Participants in Evaluation Study

(step 8).

Finally, each client executes the command to launch the specified game and joins the game server located at the address and port number decided upon by the host (step 9).

3.2 Selection Algorithm

As mentioned above, the host runs the set of client data through several steps in order to choose the best server for this particular set of clients. There are two main steps in the selection algorithm.

The first step is responsible for culling the data by identifying and removing any problematic servers. The first phase of this step insures that the server selected comes from a valid set of servers. For example, if one client fails to report a ping for a certain server because it cannot be reached from the client, this algorithm is responsible for removing the server in question from the set of possibilities. Another phase removes servers that do not have enough empty player slots to host the number of players currently looking for a game. For example, if the host is searching for a game server to support 8 clan members, a server with a maximum player value set to 6 is discarded as well as a server that has 10 out of a maximum of 16 players. The last phase prunes the server list based on the game options initially selected by the host. For example, if the game preference is for Deathmatch on the Tokara Forest map, all servers running Capture the Flag servers or the Flux2 map are discarded.

After the first step, each entry in the culled server list meets the requirements specified by the host. The second step in the game server selection algorithm is responsible for deciding which server from the culled list is the best for the players.

In order to illustrate some of the many possible algorithms for choosing the best game server, assume a host considering the best game server for eight clients, each with ping times to game servers A, B and C as shown in Table 2.

Player	Server A	Server B	Server C
1	24	74	95
2	17	62	89
3	41	100	92
4	35	51	88
5	18	84	96
6	27	44	87
7	30	122	93
8	272	71	94
Avg	58.0	76.0	91.8
Stdev	86.8	25.7	3.4

Table 2: Example Ping Times (in ms) for Eight Players and Three Servers

An algorithm that chooses the lowest average ping time will choose server A. The average ping time of 58 ms is certainly playable on average, but player 8 with a ping time of 272 is going to have a difficult time playing the game.

An extension of the lowest average ping time is to pick a maximum threshold and the game server chosen must fall under the thresholds for all clients. Assume the maximum threshold is 150, meaning that any server which returns a ping greater than 150 for any of the clients is discarded. Under these conditions, Server A will be removed from consideration and instead Server B will be selected.

Another possible solution would be to opt for fairness and choose the server which has the smallest deviation amongst the client pings. In this case, Server C with the lowest standard deviation among client pings would be selected.

Currently, our system uses the straightforward selection of the server with the lowest average ping time of all the clients. Implementation and evaluation of possibly better algorithms, as suggested above, are left as future work.

The overhead of a server selection algorithm compared with the state of the art should consider additional processing, extra network capacity and added latency. For processing overhead, with careful coding, the host can process the ping lists for each client once, building a hash table or even an array indexed by game server ID, and then removing and selecting the best server in one more pass. The overhead for this is $O(C \times S)$ where C is the number of clients that wish to play together and S is the number of game servers from which to choose. For network overhead, the host sends only a few packets to each client and the ping information gathered by each client can be aggregated and sent to the host in a few packets, so $O(C)$. For added latency, there are several additional round-trip times required from client to the host, $O(C)$, but this is negligible compared to the delay for the last client to finish pinging all game servers.

4. EVALUATION

4.1 Setup

To test the accuracy of our multiple player, game server selection system, four players from around the world volunteered to run our software client locally with the goal of finding the best Quake III server for all of them to play on simultaneously. Table 1 depicts the basic player statistics.

Around 14:00 Eastern Standard Time, each volunteer connected their client to the game server selection host running on the PC at IP 130.215.28.221 (located at Worcester Polytechnic Institute (WPI) in Worcester, Massachusetts, USA).² As detailed in Section 3, upon initiation of the four connections, the host instructed the clients to gather game

²Information on the WPI campus network can be found at <http://www.wpi.edu/Admin/Netops/MRTG/>.

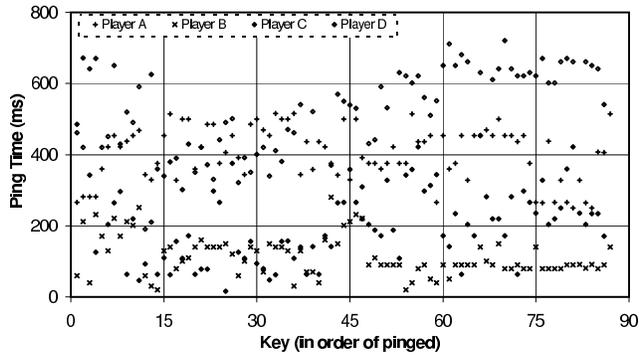


Figure 3: Scatter Plot of Game Server Ping Times

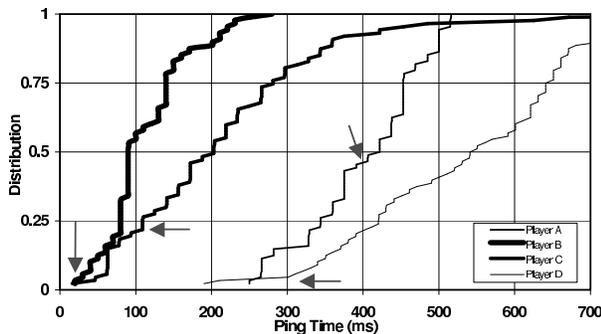


Figure 4: Cumulative Distribution Function of Game Server Ping Times

server information and send it to the host, processed the game server information, selected the best server and instructed each client to launch the game to the selected server.

The Quake III server ping times for each player were stored at the server for further analysis, as was the IP of the game server selected.

4.2 Results

Figure 3 depicts a scatter plot of the game server pings (in milliseconds) for each player, with the Key representing the ping order of the game server. The order for pinged is the same for each client since it is provided by the Quake III master server (`master3.idsoftware.com`). Broadly, the ping values vary considerably for all players, with low-end ping values under 100 ms and high-end ping values over 600 ms. There is considerable overlap in ping values for most players, indicating ping values to the same game server depends strongly upon the player location. The best server in terms of lowest average ping time is not at all obvious.

Figure 4 depicts a cumulative distribution function (CDF) of the game server ping times (in milliseconds) for each player. The game server selected by the host (the game server with the lowest average ping value) is depicted with an arrow pointing to the server location in the distribution for each player.

The overlap in ping distributions in the graph is clear,

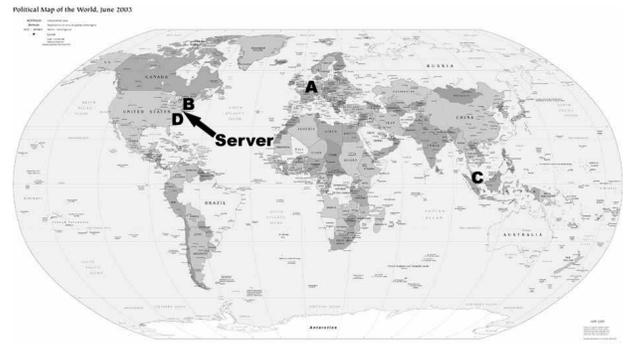


Figure 5: Map Showing Locations of Players and Selected Game Server

with player B and player C having more similar ping distributions than players A and D. This is somewhat surprising given the geographic proximity of players A and D. Most of the server ping times for player A are quite low, with over 50% being under 100 ms. Players A and D have high server ping times with even the lowest server pings being over 200 ms.

The best game server chosen for the game by the host is in the lower 50th percentile in server pings for all players. Despite player D having the highest distribution of ping times, the selected game server has a lower ping time for player D than it does for player A. The best game server in terms of lowest average ping time happens to be the overall best game server for player A. This illustrates the potential unfairness of an algorithm that only considers lowest average ping time.

Method	Average Ping	Selection Time
Manual Best	202 ms	6400 seconds
Manual First	531 ms	320 seconds
Automatic Best	202 ms	30 seconds

Table 3: Comparison of Automatic versus Manual Server Selection

Through the use of domestic and international WHOIS databases, the locations of each player’s DNS server and the chosen game server’s DNS were mapped to their respective cities. These locations were then plotted onto a world map in order to visualize the geographic separation of the players and the location of the resulting server chosen by the host. The game server selected as best resided in New York City, NY, USA. Figure 5 depicts the geographic location of the four players and their best game server.

It is difficult to compare the performance, in terms of accuracy and time, of the above approach, call it *Automatic Best*, to the manual approach of game server selection, since the time required for manual selection and the accuracy obtained varies greatly across users. However, in an attempt to illustrate the potential benefits of the proposed automatic

approach, consider two mechanisms for manual selection of a game server for multiple players. The first, called *Manual Best* uses the process described in Section 2 to manually find the server with the lowest average ping time. The second, called *Manual First*, tries to more rapidly find a game server by examining only the top game server choice for each player and selecting the server that has the lowest average ping time. Assume a particular server can be found by sorting a list and scanning in about a second and that players make no mistakes in their manual computation. Table 3 depicts a comparison of the performance for these three approaches.

The automatic approach described in this paper provides the lowest average ping time in a relatively short amount of time. Both manual methods take considerably longer than the automatic method. Even the Manual First method, arguably the fastest manual method a group of players would use, takes about ten times longer than the automatic method and selects a game server with a considerably higher average ping time.

5. SUMMARY

The increasing growth of online games has brought with it the desire for more people to play games together at the same game server. Many games have numerous game servers from which to choose and the choice of game server matters, both for type of game and for overall game performance. Unfortunately, current game server browsers only provide support for individual players, lacking the means to share information among multiple players. Players wishing to play together must manually coordinate the game server browser data via some external means, a tedious and error-prone process.

This paper presents a game server browser architecture, with corresponding system implementation, that supports game server selection for multiple players that wish to play together on the same game server. One player runs a master process as a host, while the other players connect to the host with their corresponding clients. The host instructs the clients to gather game server information from their individual client locations, gathers and processes the client data, selects the best performing game server among all the valid choices, and provides instructions for the clients to launch the game at the selected game server.

Preliminary evaluation illustrates the challenges in manually coordinating game servers and highlights how the multiple player approach to game server selection can work. Further evaluation of server ping times for multiple players may yield additional insights into overall server performance.

6. FUTURE

The current selection algorithm decides the best server based on the lowest average ping time (i.e. the fastest). More appropriate algorithms may be lowest variance in ping times (i.e. the fairest), or lowest variance in ping times with upper thresholds on a maximum tolerable ping time (i.e. the fairest with acceptable performance). The best algorithm may depend upon the specifics of the game (i.e. Quake II might require a different algorithm than Doom 3). Given the differences in latency requirements for some classes of games [8], it is even more likely the best algorithm depends upon the game genre type (i.e. a real-time strategy game, such as Age of Mythology or Warcraft III, might require a

different game server selection algorithm than a first person shooter, such as Quake III or Return to Castle Wolfenstein). In addition, the proposed algorithm could be modified to take into account the impact of the players on the chosen server, since the performance of the server could degrade if a large number of clients suddenly connected and started to play.

The proposed architecture uses a centralized solution, where the host node handles the entire server selection procedure for all the clients that wish to play together. In general, centralized solutions are not fault tolerant in that server selection will fail if the host node fails during the selection procedure. Also, the proposed centralized solution will not scale if, say, thousands of players want to play a game together (assuming a game server could support that many players). In addition, the clients need to be manually informed of the IP address of the host node, which can be cumbersome at best and does not scale if the host must communicate this IP address externally to each client.

Improvements to single player server selection were proposed by Chambers et al. in [3]. Instead of QStat, their approach uses geographic mapping and latency estimation tools to make single player server selection more accurate and efficient (in terms of network packets). Combining these single player server selection techniques with the multiple player server selection techniques presented here could be interesting future work.

7. REFERENCES

- [1] G. Armitage. An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3. In *Proceedings of the 11th IEEE International Conference on Networks (ICON)*, Sept. 2003.
- [2] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames)*, Sept. 2004.
- [3] C. Chambers, W. chang Feng, W. chi Feng, and D. Saha. A Geographic Redirection Service for On-line Games. In *Proceedings of the 11th ACM Multimedia Conference*, Nov. 2003.
- [4] S. Jankowski. QStat - Real-time Game Server Status. [Online] <http://www.qstat.org/>.
- [5] S. McCreary and k claffy. Trends in Wide Area IP Traffic Patterns: A View from Ames Internet Exchange. In *Proceedings of ITC Specialist Seminar on Measurement and Modeling of IP Traffic*, Sept. 2000.
- [6] J. Nichols and M. Claypool. The Effects of Latency on Online Madden NFL Football. In *Proceedings of the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2004.
- [7] L. Pantel and L. C. Wolf. On the Impact of Delay on Real-Time Multiplayer Games. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, May 2002.
- [8] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The Effect of Latency on User Performance in Warcraft III. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames)*, May 2003.