Measuring a Wolfenstein Enemy Territory Master Server's Response to Game Client Queries

Grenville Armitage, Carl Javier, Sebastian Zander
Centre for Advanced Internet Architectures. Technical Report 060410A
Swinburne University of Technology
Melbourne, Australia
{cjavier,garmitage,szander}@swin.edu.au

***Abstract*- The game discovery process of many network games involves game clients querying one or more master servers who hold a list of all currently active game servers. In this report we investigate the manner in which a Wolfenstein Enemy Territory (ET) master server chooses to maintain and report the list of currently active ET servers around the Internet. We note how the ordering of game servers in the master server's reply influences game server selection by clients, and show that the master server periodically re-orders the list of active game servers. Using data gathered between late 2005 and early 2006 we observe patterns of periodic re-ordering that ensure every active game server has an equal chance of being polled by game clients, regardless of the game server's location on the Internet. To reduce the probe traffic on game servers we believe some form of filtering is desirable to bias game client probing towards closer/local game servers.**

Keywords- Game Master Server, Game Server List, Network Overhead, Suitable Server

I. INTRODUCTION

This paper was motivated by earlier work on the impact of Wolfenstein Enemy Territory (ET) [1] game client query traffic on two independent online game servers [2]. One mechanism used by ET game clients to discover active game servers involves querying a master server, receiving a list of active game servers and then probing all the listed game servers for their current status. Each probed game server returns information such as current map, number of other active players and various related game-state details. Based on this information the player then chooses a game server to join – sometimes even before all game servers have responded to the game client's probes.

In [2] we noticed two things about the probe traffic from game clients from around the planet. First, the level of inbound probe traffic was independent of a game server's own popularity with players. Second, the topological origins of probe traffic seemed proportional to the population of game players in different parts of the world. In this new work we set out to test a key assumption in [2], namely that the master server does not introduce any bias into the game server probing process.

We implemented an artificial 'game client' that periodically queried the ET master server and logged the list of game servers returned for each query. By tracking the ranking of particular game servers in the master server's responses over time we made two notable observations - the master server was not introducing any particular bias in how it periodically re-ordered individual game servers, and over time every game server would be ranked at all positions in the list returned to each game client.

Due to the well-known latency intolerance of first person shooter (FPS) games [7], players tend not to spend much time on game servers that are topologically distant. In light of this, we believe it would be advantageous for clients to explicitly bias their probing of game servers towards game servers believed closer to the querying client. We hypothesise this would present a player with attractive game servers more quickly and decrease the amount of client-initiated probe traffic inflicted on distant game servers.

The rest of the paper is organized as follows: section II describes our test methodology; the results are presented in section III, and section IV concludes with comments about future work.

II. TEST METHODOLOGY

Our test methodology involved approximating the experience of a normal ET game client, and querying the master server over three different rates and periods of time. We gathered lists of active game servers at 30-minute intervals over 22 days, 60-second intervals over 4 days and 10-second intervals over 2 days. In this section we recap the actual game server discovery process, summarise the consequences for network traffic and client behaviour and then describe our query sequences.

A. *Enemy Territory Game Server Discovery Process*

Figure 1 illustrates the two main stages of game server discovery. An ET game client first contacts the master server (etmaster.idsoftware.com) by sending a UDP request to port 27950 for information about currently registered game servers (step 1). The master server then responds (step 2) with a sequence of one or more UDP packets containing the current game servers, encoded using 6 bytes for the IPv4 address and UDP port number of each registered game server[1]. (There may be multiple game servers concurrently hosted at a particular IP address, distinguished by their different UDP port numbers.)

Once the list is retrieved (step 3), the game client

[1] The Kquery [8] website provided directions on encoding and decoding the ET master server query/response packets.

begins probing each game server in sequence (step 4). The game client populates its on-screen server browser (step 5) as game servers respond with their current game information (for example, round trip time to game server, number of current players and current map). The player then choses a game server to play on from the information presented in the on-screen server browser (step 6).

We observed in step 2 that all but the last UDP response packet would have a payload of 810 bytes and contain 112 servers. The final UDP response packet would be of variable length and contain the identities of up to 112 servers. During our trials we typically saw between 26 to 28 packets in any given response, returning a list of roughly 3000 registered game servers at any given time.

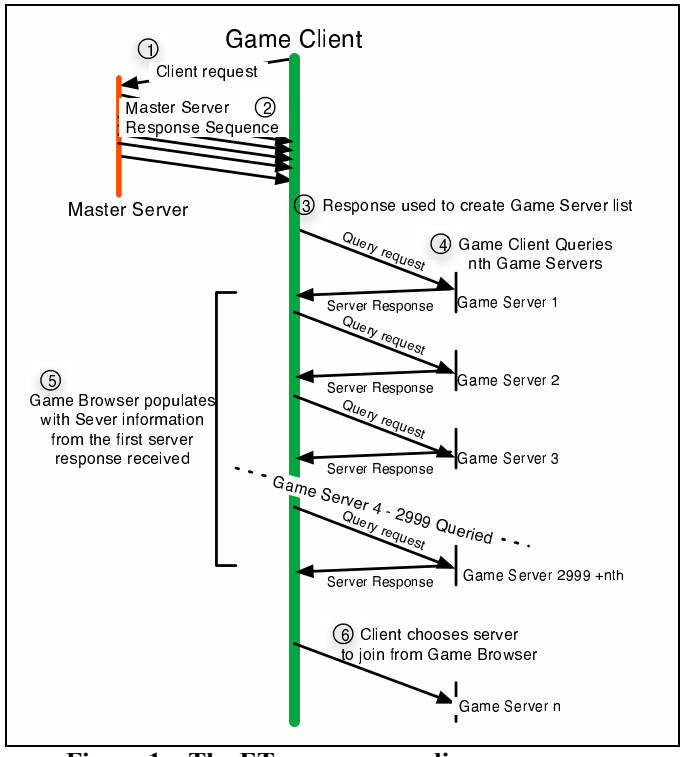

Figure 1 – The ET game server discovery process

B. *Characteristics of the discovery process*

A game client's normal server discovery mechanism has three noteworthy characteristics.

During step 4 game servers are queried in the order they were listed by the master server's response in step 2. In other words, the first listed game server in the first UDP response packet is queried first, and so on. At any given time during step 5 the player may choose to terminate the probe process and connect to a game server, even though the in-game server browser is still being populated. This introduces a slight bias against eligible game servers who are listed near the bottom of the master server's list (as the player may not wait long enough to see them[2]).

Many game servers may have high round trip time (RTT, or lag) relative to the client, and thus are unlikely to be chosen for game play [7] even though the client takes time to probe them. Consequently, the browser population stage would be expedited if the client pre-sorted the servers in step 3 to query 'local' servers first.

Game client probe traffic can contribute noticeably to overall Internet traffic volume in and out of servers regardless of the server's popularity. Our previous study [2] saw 8 gigabytes of probe traffic over 20 weeks directed at a little-used ET game server in Canberra, Australia. Over 80% of this traffic came from overseas countries whose players were unlikely to find the latency satisfying for game play. (In other words, the Canberra ET game server paid to carry probe traffic that was never likely to result in new and satisfied players.) This probe traffic from distant game clients might be substantially reduced if game clients queried 'local' servers before querying distant servers.

C. *Estimating the master server response time*

By running a series of queries in quick succession we were able to ascertain that the ET master server responds to client queries in roughly two seconds. Figure 2 shows the response sequence to three queries spaced ten seconds apart. Each query elicited a short burst of response packets within two seconds, so we could safely expect to receive all master server response packets even when sampling as quickly as six times per minute. (We were 17 hops away from the master server with an RTT of 220ms. This suggests the master server took roughly 1.7 seconds to create every response.)

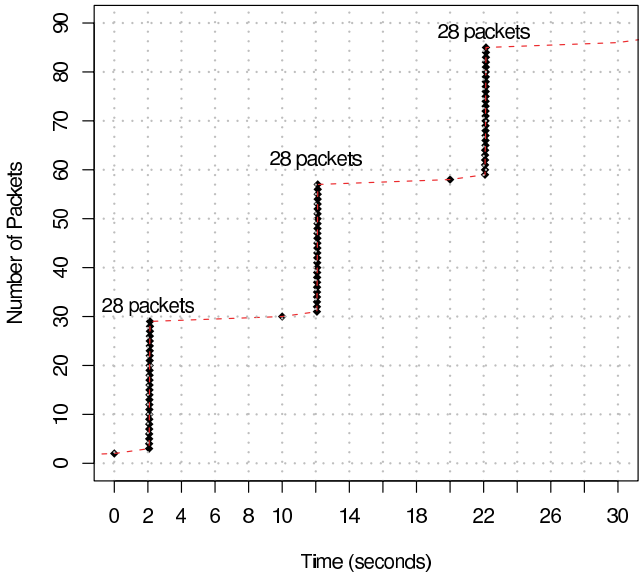

Figure 2 – Packet count versus time as ET master server responds to three game client queries 10 seconds apart

D. *Ranking servers within query responses*

We intended to primarily focus our analysis on how each registered game server's position in the master server's response list changed (or did not change) over time. Consequently we assigned each game server a

[2] A number of repeated trials using the master server query tool Qstat [6] revealed that it would take roughly one minute to probe all 3000 game servers typically offered by the ET master server at any given time.

numeric rank based on where their IP address and port number appeared in the sequence of response packets. For example, a game server whose IP address and port number appeared first in the UDP payload of the first response packet would be 'at' position one. A game server whose IP address and port number appeared 10th in the UDP payload of the 3rd response packet would be 'at' position 234, and so on.

In order to establish clear 'markers' within the master server's query responses we ran one ET game server at CAIA for the duration of our experiment. This provided us with at least one IP address:port that we knew should appear in all master server responses. (We also briefly started and stopped a 2nd ET game server at controlled times to ascertain how the master server treated newly registered game servers.)

Unfortunately the response sequence shown in step 2 of Figure 1 is intrinsically unreliable. Packets over the Internet may be lost at any time. In the absence of embedded sequence numbers there is no direct way for a client to know if it has received all the packets making up the master servers' reply. A lost packet from time to time could introduce a sudden drop in a game server's apparent rank by up to 112 positions (the maximum number of game servers in the lost packet).

As the ET game client does not compensate for packet loss we chose to simply accept the 'noise' that loss events would add to our results. (Appendix A has a discussion on possible methods to detect and compensate for packet loss across consecutive queries to the master server. However, we chose not to use any such techniques because it would not be representative of how today's ET clients experience the master server's responses.)

E. *Periodic querying over different time scales*

Our experimental query process involved a trade-off between maximising sampling accuracy and minimising network load. We intended to periodically query (sample) the master server at a rate sufficient to expose any periodic characteristics in the master server's responses[3]. However, every query generates roughly 28 UDP packets from the master server to our artificial client, so we intended to minimise this network traffic load on the master server and intervening networks.

Our initial hypothesis was that the master server might exhibit hourly, daily or weekly trends in how it reports the list of registered game servers. Thus our first experiment involved querying the master server every 30 minutes over roughly 22 days (from October 20th to November 10th 2005). This sampling interval would expose any periodic changes in the master server's response occurring at intervals of an hour or more. In Section III we refer to this is the 'long' trial.

During our first experiment we also sampled the activity levels of all registered game servers themselves every six hours. This sampling interval was considered sufficient to observe daily trends that have been reported in other literature (e.g. [2][7]). Every 6 hours we took the response list from our most recent 30 minute master server query and fed it to Qstat [6], which returned current round trip time (RTT) to, and number of active players on, each responding game server.

Subsequently we ran shorter trials to reveal periodic behaviours with far smaller intervals. On the 5th and 6th of December 2005 we issued queries every 10 seconds over 2 days. We re-confirmed certain master server behaviours between 13th and 17th of January 2006 using a less network-intensive rate of one query every 60 seconds over 4 days. As will be discussed in the next section, the increased sample rates provided valuable insights into certain periodic behaviour noticed in the master server's responses. We did not probe the individual game servers during the additional 2-day and 4-day sampling periods. In Section III we refer to these as the 'short10' and 'short60' trials respectively.

F. *Concurrent querying from diverse sources*

We made a simplistic attempt to detect whether the master server modifies its responses based on a querying client's IP address. Three artificial clients were set up in Melbourne (one on a consumer broadband connection and the other two from within our CAIA lab) and configured to query the ET master server concurrently.

Unfortunately, this test could only determine whether a querying client's precise IP address seemed to influence the master server's offered response. If the master server used a GeoIP database [5] to discriminate between clients all three of our artificial clients would appear to be in the 'same' location and thus elicit the same response. We were not able to initiate concurrent queries from entirely different countries.

III. RESULTS

In this section we will review the raw results and describe what our data reveals about the ET master server's query-response behaviour. In short, we find that over 36 minute intervals the master server moves every registered game server across the entire range of possible positions in the response list sent to querying clients. Any given client query is as likely to see a particular server at the top, middle or bottom of the list. The master server's rankings appear unaffected by the relative distance between each game server and the querying client.

A. *Review of the raw results*

Our three different trials are summarised in Table 1. Our 30-minute samples over 22 days are known as the 'Long' trial, our 60-second samples over 4 days are the 'Short60' trial and our 10-second samples over 2 days are the 'Short10' trial. Figure 3 shows a CDF how many game servers were returned in each master server response list for the Long, Short60 and Short10 trials respectively.

Responses typically returned around 3000 registered game servers. We discarded responses with less than 2500 game servers on the assumption they'd suffered substantial packet loss. For example, during the Long

[3] In practice one should sample at least twice the frequency of any periodic behaviour you might hope to observe.

trial around 3% of samples were considered anomalous and discarded because they returned responses with less than 2500 game servers. (See Appendix A for a longer discussion of detecting packet loss in master server responses.)

Table 1 - Summary of Sampling Experiments

Sample Interval	30 min (Long trial)	60 sec (Short60 trial)	10 sec (Short10 trial)
Duration	22 days	4 days	2 days
Dates	20 October to 10 November 2005	13 to 17 January 2006	5 and 6 December 2005
Samples	1,100	6,000	10,000
Dropped due to <2500 game servers	34 (3.1%)	235 (3.9%)	14 (1.4%)
Unique game servers	50,245	15,789	6,798
Game servers in 90% of all samples	2,185	2,656	2,758
Unique IP addresses	6,877	3,734	2,624

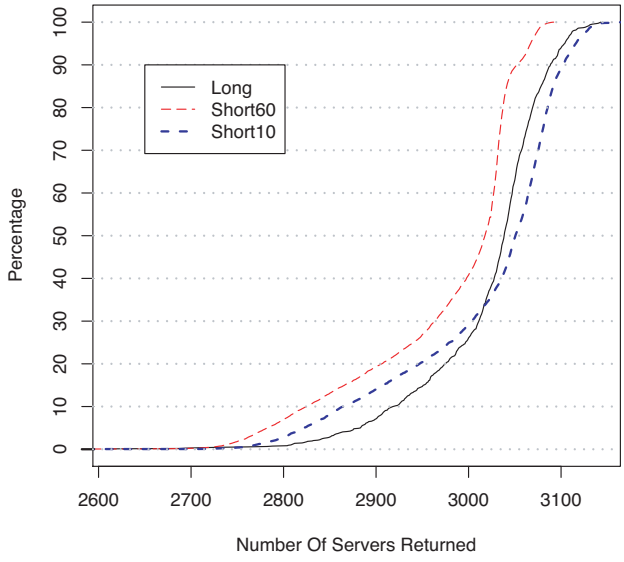

Figure 3 – CDF of number of game servers returned per query during the Long, Short60 and Short10 trials

Table 1 also shows the number of unique game servers seen over the period of each trial, and the number of unique IP addresses seen. On first glance it seems odd that, for example, the 1100 samples in the Long trial would see 50245 unique game servers and yet only 6877 unique IP addresses. The raw data revealed a core of 2185 game servers present over 90% of the entire 22-day trial period and a transient pool of game servers appearing and disappearing from one query to the next.

The set of transient game servers was dominated by a small pool of IP addresses (less than 50) that kept re-registering as new game servers over hundreds (and in some cases, thousands) of different UDP ports. We also saw low level 'noise' due to transient game servers appearing once or twice from unique IP address and port combinations and never seen again. We chose to ignore the transient servers in our subsequent analysis in this report (as the vast majority of registered game servers appeared stable across each sample period[4]).

B. *Geographic distribution of game servers*

In previous work [2] the GeoIP database [5] was used to map client IP addresses to approximate geographic origins. We saw a distinct bias in the geographic distribution of game clients querying two ET game servers under our control. A majority of game client queries came from Europe and the USA.

Again using the GeoIP database, Figure 4 shows the geographical distribution of game servers reported by the master server during our 22 day 'long' trial. The dotted line encompasses the main five European-region countries, revealing the significant level of game servers located in Europe. The results in [2] and Figure 4 seem to agree with our intuition that the density of both game servers and game clients in particular geographic regions would be correlated. Where there are lots of clients there are likely to be lots of people willing to run game servers (and vice-versa).

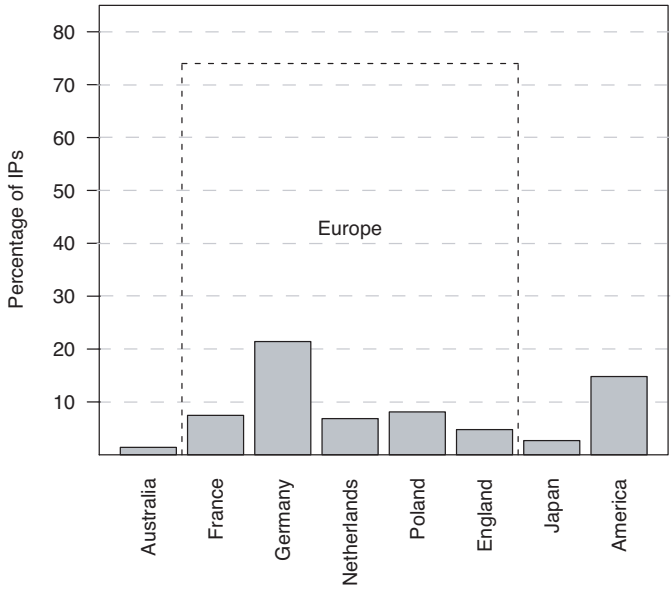

Figure 4 – Geographical distribution of ET game servers during the Long trial

C. *Weekly and daily trends in total number of game servers and levels of game server activity*

Figure 5 shows a number of details relating to the long trial and the probing of registered game servers every 6 hours. We average together the results from each 6-hour interval from each of the three weeks to reveal possible weekly or daily trends.

[4] The dynamic behaviour of transient server registrations will be analysed in a later CAIA Technical Report.

The average number of registered game servers returned by the master server every 6 hours seems to sit around 3000 and change very little across the week. (The maximum and minimum bars shown for the "Number of Servers Returned" reflects the range of values returned during 30-minute queries over the preceding 6 hours.)

Using the Qstat probes every 6 hours we can also see the daily fluctuations in the number of game servers having one or more players ("Servers with players"). The most active times are in the early morning here in Melbourne (in other words, late afternoon and evening in Europe). This is unsurprising considering the geographic distribution of game servers revealed in Figure 4.

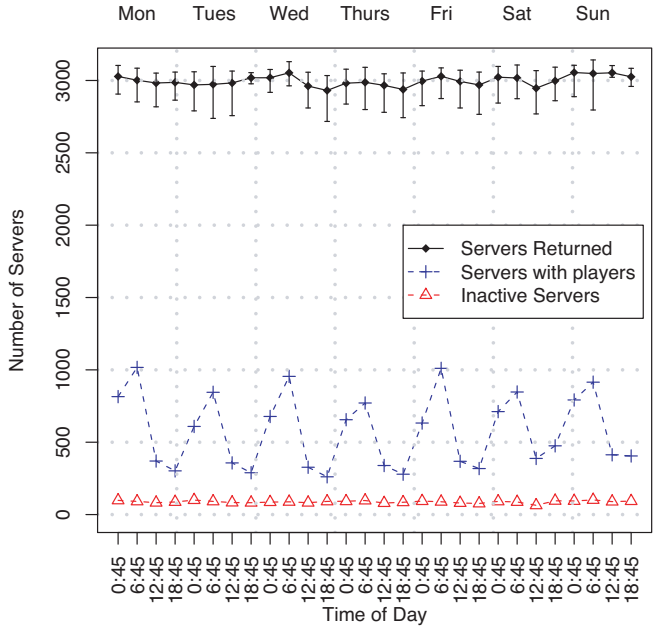

Figure 5 - Weekly trends using 6-hour samples of game server activity

Interestingly, the "Inactive Server" data points show that roughly 90 registered game servers are inactive (giving no response to our Qstat probes at all) during each 6-hourly probe. Although these inactive servers are registered with the master server, a regular game client would be unable to fully populate its in-game server browser with information on these 90 game servers.

D. *Distribution of Distribution of Game Servers on the list (Long Term Sampling Results)*

Our first test for bias in how the master server ranks individual game servers involved looking at the distribution of RTT to each game server in the master server's response list. Figure 6 is a scatter plot of each game server's RTT (measured from our artificial client) versus its rank. These RTT values are derived from a single 6-hourly probe of all game servers using Qstat.

There appears to be no particular relationship between a game server's RTT and its position in the list returned by the master server. Interestingly the scatter plot clearly shows three distinct regions of game servers as seen from a client based in Australia. Previous related work on RTT distributions of ET clients measured from Melbourne [9] suggests the lower RTT ranges (0 - 100 ms) are most likely to be local game servers within Australia and surrounding countries. The middle ranges of RTTs (180 - 270 ms) are likely to be game servers from America and the servers clustered above 300ms+ are most likely to be based in Europe.

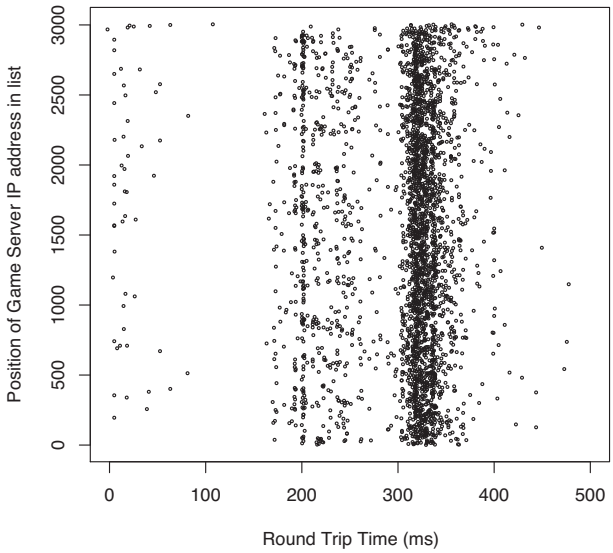

Figure 6 – RTT vs. Game server rank after one query during the Long trial

E. *Distribution of game server rankings over time*

Our second test for bias in game server rankings involved tracking the rank assigned to three specific servers over the long (22 day) trial. We tracked our own local 'CAIA' server and two other servers known to be in America and Germany respectively.

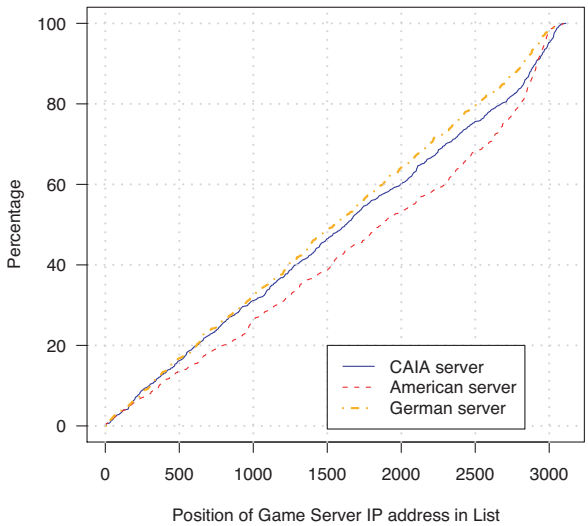

Figure 7 – CDF of game server rank during the Long trial

Figure 7 shows a cumulative distribution function for the rank assigned to each game server over all of the long trial's 1100 master server response lists. Over long periods of time (and multiple queries) each game server's likely rank in any given query response list appears almost uniformly distributed across all possible rankings.

Closer inspection (not shown here) revealed that each server's rank jumped around unpredictably (rather than increase or decrease gradually) from one query to the next. Either the master server was deliberately randomising the rankings of game servers in every query response, or there was a periodic re-ordering whose period was smaller than 60 minutes. This uncertainty led us to perform the short10 and short60 trials.

F. *Periodicity in the master server's response list*

Figure 8 shows the ranking of three game servers over a 100-minute segment of the short60 trial. Each game server's rank periodically cycled from beginning to end of the master server's response list with a period of roughly 36 minutes. In Figure 8 we tracked three game servers who were near the beginning (server at rank 1), middle (server at rank 778) and end (server at rank 2564) of an early response list[5].

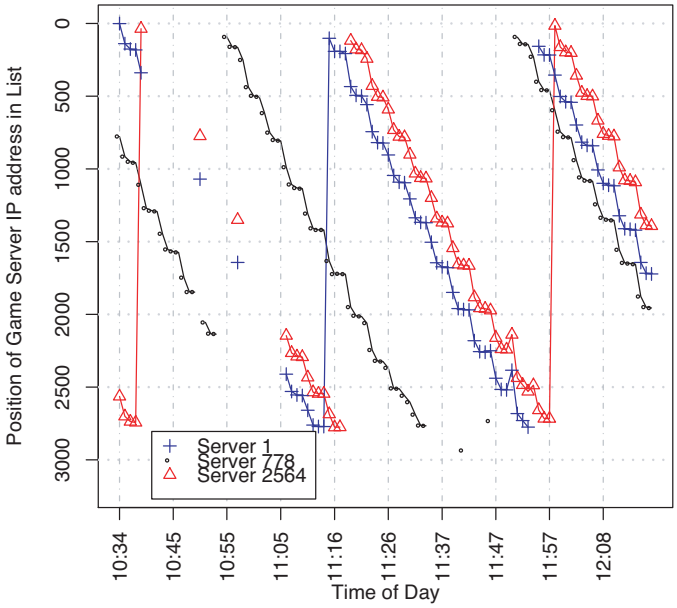

Figure 8 –Ranks of three game servers during a fragment of the Short60 trial

(Although not shown, we observed similar periodic behaviour in the rankings of other game servers. When a game server first registers with the master server it is ranked at the beginning of the response list and then begins the periodic migration from beginning to end along with all other registered game servers.)

Based on the short10 trial Figure 9 provides a close-up view covering four unrelated game servers over 15 minutes with queries every 10 seconds. Each 36-minute cycle is made up of consecutive, non-linear bursts of downward migration in ranking[6]. Each of these short bursts takes roughly 3 min 40 seconds, with an asymptotic shape that is as-yet unexplained.

[5] The lines are broken in places where particular servers were not consistently present. This provides some indication of the uncertainty a player's client would experience when querying for available game servers.

[6] As with Figure 8, some game servers came and went during the observation period.

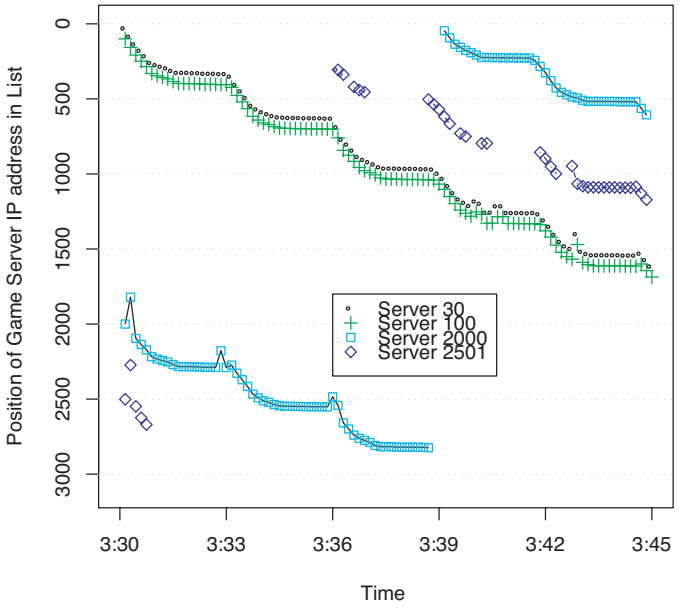

Figure 9 – Ranks of four game servers during a fragment of the Short10 trial

G. *Querying from diversely located clients*

Our final attempt to find bias involved concurrently querying the ET master server from three separate artificial clients located at different IP addresses (as described in section II.F). Figure 10 shows the rank of a particular game server as seen from three different clients who issued concurrent queries every 10 seconds over a three minute period. Clients 'caia1' and 'caia2' were located on the same 136.186.299/24 subnet, whilst 'homePC' was located on a totally different ISP within the same metropolitan area (Melbourne, Australia).

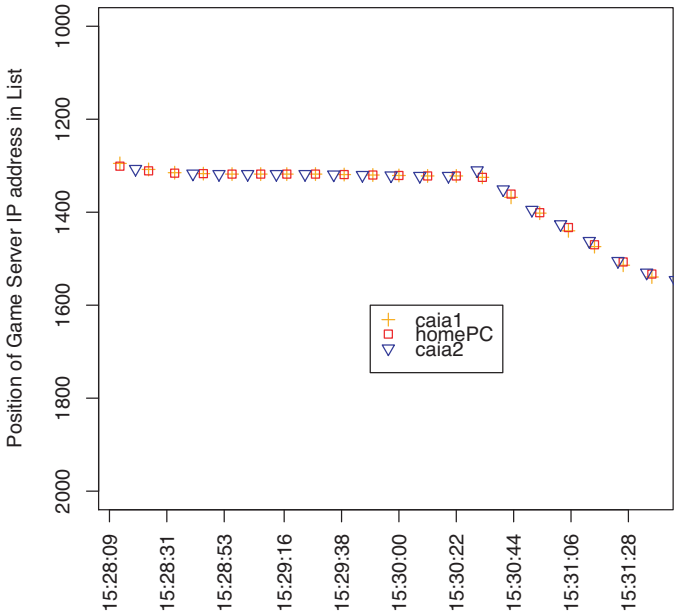

Figure 10 – Rank of one game server while being queried every 10 seconds from three separate clients

It seems reasonable to conclude that the master server is not introducing any offset to rankings based on the specific IP address. However, we cannot rule out the possibility that a client's apparent country of origin

might be used to permute the master server's rankings. Unfortunately we were not able to run concurrent client queries from hosts in different countries in order to rule out this possibility.

IV. GETTING 'STUCK' AT THE BOTTOM OF THE LIST

Most of the time game servers experience the periodic cycling of their rank according to the patterns shown in Figure 8 and Figure 9. However, from time to time an active game server would seem to get 'stuck' near the bottom of the master server's list. An example of this phenomena happening to the CAIA Server is shown in Figure 11 (with other servers also shown unaffected at the same time) and in close-up in Figure 12 (showing only the CAIA Server's rank).

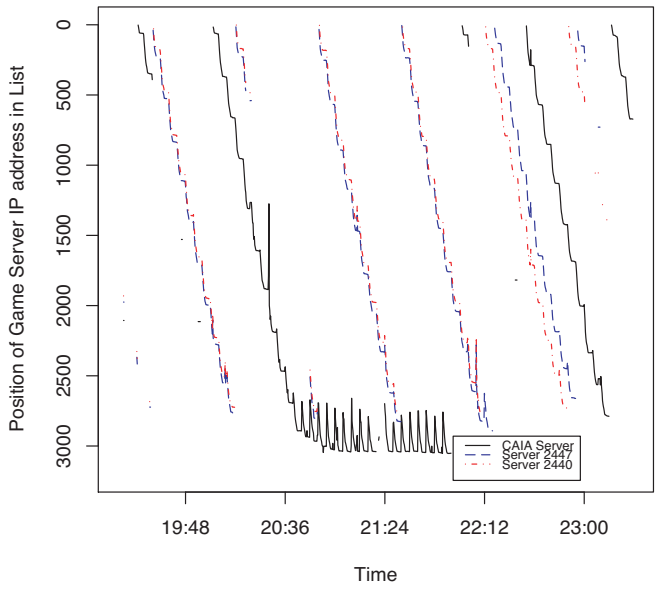

Figure 11 – CAIA's own server getting 'stuck' at the bottom of the master server's response list while other game servers continue the regular 36-minute cycle

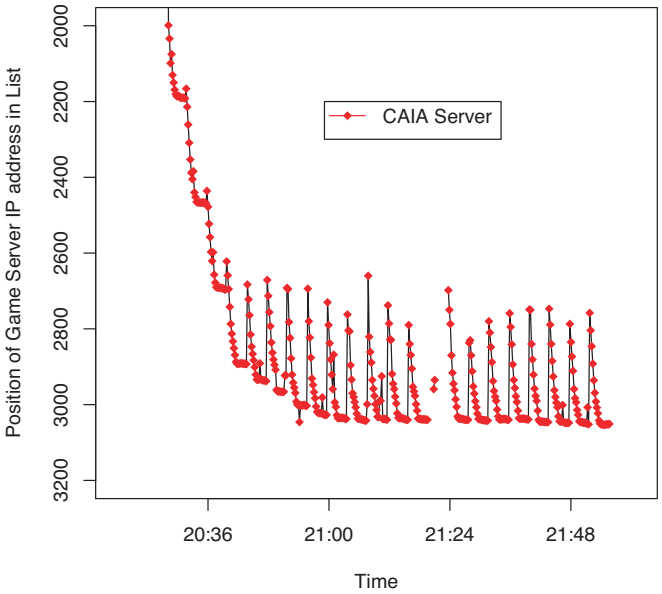

Figure 12 – Close-up of CAIA's own server getting 'stuck' at the bottom of the master server's response list

Getting 'stuck' down the bottom of the ranking is a disadvantage for game servers trying to attract players. They will be probed and presented to players only after all the higher ranked servers are presented, possibly losing an opportunity to catch a potential player's attention.

We do not yet have an explanation for this phenomenon. Fortunately, Figure 7 suggests no significant long-term influence or bias is introduced into the distribution of rankings to any given server.

V. CONCLUSION AND FUTURE WORK

This short paper was motivated by the following goal: test whether a Wolfenstein Enemy Territory (ET) master server introduces bias into the ordering of registered game servers that are returned when queried by ET game clients. Any such bias would influence how frequently game clients around the world might probe particular game servers when players trigger the in-client server browser function. We wished to validate the assumption in previous work [2] that there was no particular bias.

We queried the ET master server every 30 minutes for 22 days, every 60 seconds for 4 days and every 10 seconds for 2 days. From the master server's responses we showed:

- The majority of ET game servers are in Europe (consistent with previous work showing the majority of clients were in Europe).
- The rank (position) of any given game server in the master server's response list periodically moves from top to bottom every 36 minutes.
- The rank of each game server within the response list appears to be unaffected by the topological relationship between each game server and the querying client.
- No particular bias was observed keeping particular game servers closer to the top of the master server's list. Over a long period of time every game server is equally likely to appear at any possible rank in the master server's response.

A follow-up experiment suggested by these results is to pre-sort the master server's response list so that game server's "close" to the querying client are ranked nearer the top of the response. This may reduce the time it takes for the client's in-game server browser to present an attractive game server on which to play (on the basis that lower latency is more attractive all other things being equal). If players promptly select a game server from near the top of the browser's list there would be an associated reduction in unnecessary client probe traffic to distant game servers further down the list. We intend to explore implementations of such a scheme, and quantify the impact on network traffic.

REFERENCES

[1] "Wolfenstein Enemy Territory", http://enemy-territory.com, (viewed on 10th October 2005.)

[2] S.Zander, D.Kennedy, G.Armitage, "Dissecting Server-Discovery

Traffic Patterns Generated By Multiplayer First Person Shooter games", ACM NetGames 2005, NY, USA, 10-11 October, 2005

[3] Centre for Advance Internet Architecture, http://caia.swin.edu.au (viewed on 25th July 2005)

[4] GrangeNet, http://www.grangenet.net/advancedcommunications/wp/sim/index.html (viewd on 30th August, 2005)

[5] GeoIP, http://www.maxmind.com/ (view on 25th August, 2005)

[6] "Qstat Real-time Game Server Status", http://www.qstat.org (viewd on 10th October, 2005)

[7] G.Armitage, "An Experimental Estimation of Latency Sensitivity In Multiplayer Quake 3", 11th IEEE International Conference on Networks (ICON 2003), Sydney, Australia, September, 2003

[8] Kquery, http://dev.kquery.com/ (viewed on 6th October, 2005)

[9] G.Armitage, C.Javier, S.Zander, "Client RTT and Hop Count Distributions viewed rom an Australian 'Enemy Territory' Server", CAIA Technical Report, 6th February, 2006

[10] ServerSpy, www.serverspy.net (viewed on 28th October, 2005)

[11] FreeBSD home page, http://www.freebsd.org/ (viewed on 25th August, 2005)

[12] D.Kennedy, B.Tyo, "Configuring a ' Wolfenstein: Enemy Territory' Server for Online Play", CAIA Technical Report, August 2004

APPENDIX A. PACKET LOSS IN MASTER SERVER RESPONSES

An ET client cannot easily detect whether any given master server response list is in fact the complete list of registered game servers held by the master server at the time of the query. Packet loss is a fact of life in the Internet, and a master server response of N packets could easily arrive with N-1, N-2 or even less packets. As noted in the report, packets 1 through (N-1) contain IP address and port number pairs of 112 game servers while packet N contains from 1 to 112 game servers.

In this report we made no attempt to identify if packet loss was influencing our results. Rather, we chose to evaluate the sever distributions that would be experience by regular game clients regularly querying in the master server. However, a number of possible techniques could be deployed to infer packet loss. Consider a master server response of N packets. A loss of packet X (where 1<=X<N) would have two consequences:

- Exactly 112 game servers would temporarily disappear.
- All game servers between packet X+1 and packet N would have their apparent rank jump 112 positions towards the 'top' of the list (at rank 1).

A loss of packet N would have slightly different consequences:

- Between 1 and 112 game servers would temporarily disappear
- The 'last' packet received from the master server would contain precisely 112 game servers.

When sampling the master server repeatedly over short periods of time we might retrospectively infer packet loss events by looking for transient fluctuations in the number of packets returned and the total number of game servers returned.

For example, packet loss always causes a drop in the number of packets and number of game servers returned (relative to the previous, complete master server response). However, the number of packets may also drop if the number of registered game servers legitimately happened to drop across a packet boundary (e.g. from 113 to 110, or 1120 to 1115) between client queries. One possible sign that a drop is caused by packet loss could be when the number of packets drops and the number of game servers drops by 112 (or more) at the exact same time.

Naturally, this could also be cause by a sudden, legitimate loss of 112 (or more) game servers from the master server's list. We can be more certain the original drop is due to transient packet loss if the very next query returns a matching *increase* in the number of packets (and game servers) returned by the master server. Thus, a transient drop-then-jump of one or more packets (and 112 or more game servers) between response R1, R2 and R3 would be an indication that R2 had been impacted by transient packet loss.

If we lost packet X (where 1<=X<N) we might also strengthen the inference of packet loss by noting if multiple game servers had their rank suddenly drop-then-jump 112 positions between responses R1, R2 and R3. Note that a drop alone is not sufficient, and this only works if we monitor game servers in packets X+1 to N. Although we do not know which packet was lost, the symptom 'a transient drop-then-jump of one or more packets (and 112 or more game servers)' suggests the lost packet X in response R2 was indeed in the range 1<=X<N. In such cases we can strengthen the inference of packet loss by noting whether the game servers in found in the last packets of both R1 and R3 seemed to jump in rank by 112 positions in R2.

We might even infer where in R2 the packet loss occurred, on the assumption that the trailing packets X+1 to N in R1 and R3 will carry much the same list of game servers as the equivalent set of trailing packets in R2. Keep stepping back towards the start of R2 until the trailing game server lists significantly differ – you'll have located the likely point of packet loss.

If the last packet is lost there is also transient drop-then-jump in the number of packets. However, the number of listed game servers drops and jumps by less than 112 (since the last packet may contain as few as one extra game server). In this case we can strengthen the inference of packet loss by noting whether the game servers listed in the last packet of R1 and R3 are (a) basically the same and (b) totally absent from the entire list of game servers in R2.

The techniques discussed here can be applied to previously captured packet traces of client query and master server responses. From this we might calculate the incidence of packet loss on master server response lists over time.