Topological Optimisation for Online First Person Shooter Game Server Discovery

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

***Abstract*- Most online first person shooter (FPS) games require clients to discover game servers through a two-step process. The client initially queries a well-known master server for a list of currently registered game servers. Each game server is then probed in the order they were returned by the master server. It may take quite some time to discover playable game servers (within a tolerable round trip time of the client). We investigate the need for, and benefits of, explicitly re-ordering a master server's reply list so the client is more likely to probe closer servers before more distant servers. We use data gathered from the Wolfenstein Enemy Territory (ET) master server gathered in late 2005 and early 2006 to assess such optimisation. We conclude that such re-ordering will reduce unwanted probe traffic on game servers and improve player experience.**

Keywords- Game Master Server, Game Server List, Network Overhead, Playable Server, First Person Shooter, Optimisation

I. INTRODUCTION AND RELATED WORK

Internet-based multiplayer First Person Shooter (FPS) games have become significantly popular in the past 5+ years. Game servers are hosted by internet service providers (ISPs), dedicated game hosting companies and individual enthusiasts. Examples of FPS games include Quake III Arena, Half-Life Counterstrike, Wolfenstein Enemy Territory, and Half-Life 2. FPS game servers typically host from less than 10 to around 30+ players and there may be hundreds and thousands of individually operated game servers active on the Internet at any given time [1]. Locating a playable game server is a key challenge for would-be players.

The notion of a 'playable' multiplayer FPS games is influenced by the number of players already on the game server, the particular map being played, the specific game rules, and the round trip time (RTT) between game server and client. Most FPS games use a similar two-step process to gather this information [1]. First, a game client queries a master server unique to the particular game (pre-configured into the game client's software). The master server returns a list of hundreds or thousands of IP addresses and port numbers representing game servers registered as currently active. The client then steps through this list, probing each listed game server for information about map type, game type and number of players (typically a brief UDP packet exchange). This probe process also estimates the client to server RTT at the time of the probe. All this information is presented to the player as it is gathered, who then selects a game server to join.

Previous work has shown a constant 'background noise' of probe traffic to active game servers as clients from all around the planet start and stop every hour of the day [2]. Registered game servers experience a non-negligible base level of traffic regardless of their actual popularity, traffic for which the server operator ultimately pays. (For example, in [2] two Australian-based FPS game servers each experienced 8 gigabytes of public Internet probe traffic over a 20-week period, even though one server was vastly more popular with game players than the other. Over 80% of the probe traffic came from overseas countries whose players were unlikely to find the latency satisfying for game play.)

From a player's perspective this technique can lead to tedious waiting periods (tens of seconds or minutes) as the client probes each IP address returned by the master server. If a playable server appears early in the master server's list, the player may instruct the client to terminate the probe process and begin playing immediately. If not, the client may need to probe all available game servers before the player finds a server with acceptable RTT and game-play conditions.

FPS players tend not to spend much time on game servers that are topologically distant (tolerance tends to drop off noticeably as RTT heads over 180-200ms [3][4][5][6][7][8]). Thus we hypothesise that the process of discovering playable servers would be improved if clients probed 'closer' game servers before more 'distant' game servers. This would enable quicker identification of game servers within a tolerable RTT, expediting the player's decision to terminate the probe process and begin playing on a chosen game server. We further hypothesise that earlier termination of the probe process by clients around the world would reduce probe traffic loads on game servers who are distant from most clients.

Our work differs from previously published work on redirecting players from one game server to a 'closer' game server based on inferring geographic locality from client IP addresses [9]. We aim for the client itself to expeditiously find closer (and more playable) servers.

We use data gathered from the Wolfenstein Enemy Territory (ET) [10] master server in late 2005 and early 2006 to illustrate the need for, and likely benefits of, our hypothesis. Our paper is organized as follows: section II describes our characterisation of the ET master server's ranking process, section III presents the characterisation results, section IV explains our proposed optimisation in light of these results and section V concludes with comments about future work.

II. Characterising The Enemy Territory Server's Ranking Of Registered Game Servers

We created an artificial ET game client based in Melbourne, Australia, and issued repeated queries over time to the ET master server in order to track the ranking of game servers in consecutive replies.

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 responds (step 2) with a sequence of one or more UDP packets listing the currently registered game servers (6 bytes each for the IPv4 address and UDP port number). (Different UDP ports are used when multiple game servers exist at the same IP address.)

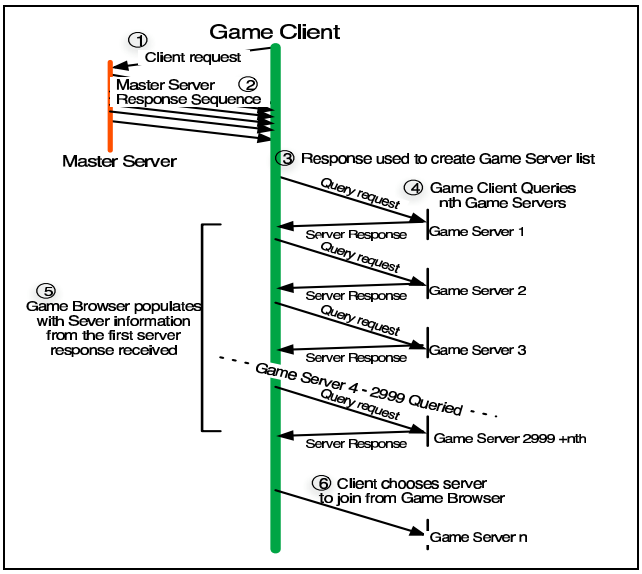

Figure 1 – The ET game server discovery process

As the list is retrieved (step 3), the game client 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 the response was a back-to-back burst of UDP packets roughly 2 seconds after we issue each query in step 1. (Approximately 220ms of that delay was the RTT between our measurement point and the master server.) 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. 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.

As a modest optimisation actual ET clients issue the first 16 queries almost immediately in step 4, without waiting for replies. It then issues new queries as replies come back for previous queries, with no more than 16 queries outstanding (unanswered) at any one time. This speeds up the discovery process while minimising load on the client's access link (minimising degradation of each query's RTT estimate).

At any time during step 5 the player may choose to terminate their client's 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.

B. *Ranking servers within query responses*

For our purposes each game server is assigned a numeric rank based on where their IP address and port number appeared in the sequence of response packets. A game server whose IP address and port number appeared first in the UDP payload of the first response packet is given a rank of one. A game server whose IP address and port number appeared 10th in the UDP payload of the 3rd response packet is given a rank of 234, and so on.

We ran a local ET game server to provided us with at least one IP address:port that we knew should appear in all master server responses. By tracking the rank of this server over time we could ascertain any trends or bias in the ET master server's behaviour.

C. *Periodic querying over different time scales*

In order to evaluate possible periodicity in the master server's behaviour we ran three separate trials with three different query intervals. Our 'Long trial' involved querying the master server every 30 minutes over roughly 22 days (from October 20th to November 10th 2005). Our 'Short10' trial involved queries every 10 seconds over 2 days (5th and 6th of December 2005) and our 'Short60' trial involved queries every 60 seconds over 4 days (between 13th and 17th of January 2006). During the Long trial we also used Qstat [12] every 6 hours to find the current round trip time (RTT) to, and number of active players on, each responding game server from the most recent master server query. The 6-hour sampling interval allowed us to observe daily trends reported in other literature (e.g. [2][6]). We did not probe the individual game servers during the short10 and short60 trials.

III. Interim Results

In brief: 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*

Table 1 summarises our results for the Long, Short60 and Short10 trials, including the number of unique game servers and unique IP addresses seen during each trial. On first glance it may seem odd that, for example, the 1100 samples in the Long trial would see 50245 unique game servers and yet only

6877 unique IP addresses. In fact, 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 additional game servers appearing and disappearing from one query to the next.

Table 1 - Summary of Long, Short60 and Short10 trials

Sample Interval	30 min (Long trial)	60 sec (Short60 trial)	10 sec (Short10 trial)
Duration	22 days	4 days	2 days
Dates	20/10/05-10/11/05	13/1/06-17/1/06	5/12/05-6/12/05
Samples	1,100	6,000	10,000
Unique game servers	50,245	15,789	6,798
Servers in 90% of all samples	2,185	2,656	2,758
Unique IP addresses	6,877	3,734	2,624

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.

For all three trials most master-server responses returned around 3000 registered game servers, with 90% of all responses returning between 2800 and 3100 game servers. (Approximately 3.1%, 3.9% and 1.4% of the Long, Short60 and Short10 trial responses were discarded for returning less than 2500 game servers – presumed to be evidence of substantial packet loss.)

Our six-hourly Qstat probes of every registered game server revealed two things. The reply packet from each game server ranged from 258 to 403 bytes long, with median and mean both ~300 bytes. At any one time roughly 90 registered game servers were inactive. A regular game client would try but be unable to fully populate its in-game server browser with information on these inactive game servers.

Slow responses impact an ET client's ability to benefit from parallel queries. Experiments using a real ET client showed that on average the ET client probed 8 game servers at a time, even though it begins with a burst of 16 queries. It took roughly a minute to probe the 3000 game servers provided by the master server.

B. *Geographic distribution of game servers*

The GeoIP database [11] is a useful tool for mapping IP addresses to their approximate geographic origins (the free 'GeoCountry Lite' version provides country-level resolution). Using GeoLite Country, Table 2 shows the coarse geographical distribution of game servers reported during a single Qstat query in the Long trial. Significant numbers of ET game servers are located in Europe, with a modest 554 servers in the USA and only 51 in Australia. 45 other countries make up the rest of the list. (A similar distribution of ET clients is reported in [2].)

Table 2 Top 10 Countries in a single six-hourly Qstat probe ranked by number of registered game servers

Germany (943)	United States (554)	Netherlands (312)	United Kingdom (209)	France (147)
Poland (83)	Finland (60)	Czech Republic (58)	Australia (51)	Japan (50)

C. *Distribution of Game Servers versus RTT*

Figure 2 is a scatter plot of each game server's RTT (measured using Qstat) versus its rank in a single master server response during the Long trial. There is no particular relationship between a game server's RTT (to the client) and the game server's rank in the master server's response. The master server is not ordering its responses based on the querying client's probable location on the Internet. Similar RTT vs. rank distributions were found in all the 6-hourly probes during the Long trial.

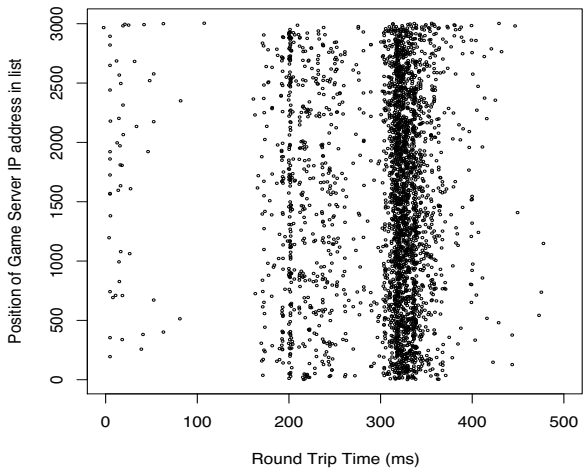

Figure 2 – RTT vs. Game server rank from one query during the Long trial

Figure 2 also clearly illustrates the problem facing ET players in Australia. Of 2981 actual servers seen in this 6-hour sample, 2296 servers are over 300+ms away (largely European, and completely unplayable). The 33 servers sitting between 160ms-179ms or the 597 servers clustered between 180ms-299ms would provide marginal game play. Only 55 game servers (51 Australian, 4 New Zealand) are less than 120ms away from our client. Yet an Australian player must probe all 2981 game servers before their server-browser covers all potentially playable servers.

D. *Distribution of game server rankings over time*

Figure 3 is a CDF of the rank assigned to three different game servers over the Long trial. We tracked our own game server ('CAIA') and two other long-lived game servers located in America and Germany respectively. 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.

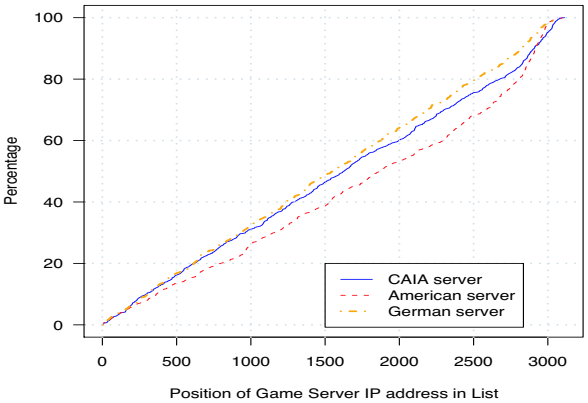

Figure 3 – CDF of game server rank during the Long trial

Closer inspection of the Short60 and Short10 trial results revealed that the ET master server cycled every game server's rank from first to last in the response list approximately once every 36 minutes. A newly registered game server would be injected at the top of the list (rank of 1) and then begin migrating to the bottom of the list over the next 36 minutes before reappearing at the top of the list again.

IV. A Proposed Optimisation

Client-side game-server discovery will occur faster if the client probes 'closer' game servers first.

A. *Theoretical improvements*

First consider an idealised scenario (*scenario A.1*):

- All N available game servers are uniformly distributed between 20ms and 350ms from the querying client (based on typical client access links adding 10-20ms).
- Every game server's rank (from one query to the next) is uniformly distributed between 1 and N (Figure 3)
- Game servers are probed sequentially, the median response is 300 bytes long and game servers over ~180ms away are 'unplayable'

Only 48.5% (160/330) of N are playable servers. Given the master server ordering a client must query all N game servers to discover these playable servers. Each query takes at least (20+350)/2 = 185ms (considering only the RTT) for a total of N*0.185 seconds. Probing 3000 game servers sequentially would take 555 seconds. 16 queries in parallel (section II.A) would take ~35 seconds under ideal conditions and trigger ~878Kbytes of inbound traffic to the client.

Search time would improve if the master server presented game servers in order of distance from the querying client (*scenario A.2*). Knowing this, the player can terminate their search upon seeing a game server over 180ms – thus probing only the first 48.5% of the returned game servers at an average of (20+180)/2 = 100ms per query. Over 3000 servers there would be 1455 playable game servers probed in just 145.5 seconds (or 9 seconds if 16 queries are issued at a time under ideal conditions). This optimised server discovery process would result in only ~426Kbytes of inbound traffic.

Actual Australian game clients do not see a uniform distribution of available servers. Only 2.95% of all servers are less than 180ms away (and plausibly playable). Let us assume our client sends 16 queries in parallel and playable servers are distributed uniformly between 20ms and 180ms (*scenario A.3*). The 89 playable servers would be probed within half a second and trigger only ~26Kbytes of inbound traffic. There are clear benefits to be had from optimising the search for playable game servers.

B. *An address-independent client-side technique*

Our proposal is for a client-side algorithm that samples the master server's list to construct an estimate of likely RTTs to different countries, and then re-orders the game servers along country lines (*scenario B.1*). This technique works wherever the client is located on the Internet, and does not require any local configuration of location or IP address information - an advantage given the prevalence of NAT (network address translation) [14] on consumer home gateways.

First the client takes the existing master server response list and groups game servers together by country (e.g. using a GeoLite Country list embedded in the game client [11]). One game server is selected at random from each represented country, and each of these selected game servers is probed in a random order. The RTT to each of these initial game servers is then used to rank the countries they represent (and by implication the other game servers 'within' each country). The game client then begins probing all game servers in order of their country's new rank. In the absence of any additional hints, game servers within a given country would be probed in random order.

This approach works well for Australian ET clients. Ranking the 55 countries represented in Figure 2 requires 55 initial probes and 55 replies (~16Kbytes of inbound traffic). Australia and New Zealand will be ranked first and second, and the client will proceed to issue 53 more probes (to cover all 55 Australian and New Zealand game servers). Further assume the probe process is terminated once the client begins displaying RTT values that the player considers to be 'too high'. The whole process requires only 110 probes and ~32Kbytes of inbound traffic. Even if we pessimistically allow ~185ms for all 110 probes, this would take only 1.3 seconds (assuming 16 probes in parallel).

Clients in countries with high concentrations of local servers will see proportionally less benefit, since they see proportionally more 'playable' servers (based on RTT) when using the master-server's lists as-is. At worst such clients will see performance converging to the un-optimised case.

Game servers also stand to benefit by a reduction in probe traffic from clients who are unlikely to ever play on them. For example, European ET clients would place Australia (and hence all Australian game servers) at the end of their search after probing one Australian server at random. This could eliminate up to 80% of the probe traffic seen in [2] from

overseas clients (reducing 8Gbytes of probe traffic over 20 weeks to just 1.6Gbytes).

Table 3 summarises the results of our ideal optimisations (scenario A) and proposed optimisation (scenario B).

Table 3 Summary of ideal (A) and proposed (B) optimisations

Scenario	A.1	A.2	A.3	B.1
Time to probe (16 at a time)	35 sec	9 sec	0.5 sec	1.3
Inbound traffic	878KB	426KB	26KB	32KB

V. FUTURE POSSIBILITIES

There are a number of future refinements.

Probing a single, randomly selected game server to rank each country may be quite misleading for larger countries with hundreds of servers and diverse internal topology. A practical refinement would be to cluster game servers inside each country using an indirect indication of possible topological locality. Within each country, subdivide the game servers into groups based on sharing e.g. /8 or /16 IPv4 routing prefixes. Randomly pick and probe one game server from each <country,prefix> group, and then rank all game servers according to the nominal RTT of their group. Order game servers randomly within their group and proceed with the client probe sequence as before.

Even using a /8 prefix leads to a lot of initial probes. For example, there are 15, 33 and 17 unique /8 prefixes seen under Germany, United States and Netherlands respectively. There are 316 <country,prefix> groups using /8 prefixes across all 55 countries seen in Figure 2 (hence 316 initial probes before the client can begin re-ordering the master server's response list).

There are two underlying assumptions: topological locality implies shorter path lengths (and hence RTT), and geographic indicators and routing prefixes correlate with topological locality. The latter becomes weaker for shorter prefixes, particularly as disjoint regions of the planet may share a /8. Thus we do not advise grouping purely on <prefix>. Even though grouping game servers by shared /8 would give us only 81 groups to initially probe, the choice of random server within each /8 group is likely to be even less representative of the entire group's RTT than grouping on country alone.

In principle the master servers themselves (or a transparent proxy) can also optimise the response lists. Since they know the querying client's IP address they can pre-order the returned game server list in order of likely locality to the client (either using country lists like GeoLite Country, or making assumptions based on shared routing prefixes). However, the master server would be performing per-client/per-query optimisations on the fly for thousands of clients per hour. This may be an unacceptable processor load. Thus we prefer to consider client-side techniques (where the load is distributed and localised).

VI. CONCLUSION

Existing FPS game server discovery can be a slow process generating hundreds of kilobytes of mostly useless network traffic. In particular, clients located over 180ms from the majority of registered game servers incur a substantial penalty during the discovery process. Using statistics gathered from a Wolfenstein Enemy Territory master server we illustrate the benefits of novel client-side technique for re-ordering the game server lists provide by the master server. A typical discovery sequence causing ~878Kbytes of inbound traffic over 35 seconds could be reduced to less ~32Kbytes of inbound traffic in under two seconds for Australian-based players. In addition, game servers will see noticeably reduced probe traffic from clients. The server-discovery process is similar for most popular FPS games [1], so we believe our proposed client-side optimisation is similar applicable and beneficial to other FPS-style online games.

REFERENCES

[1] G.Armitage, M.Claypool, P.Branch, "Networking and Online Games - Undertanding and Engineering Multiplayer Internet Games," John Wiley & Sons, UK, April 2006 (ISBN: 0470018577)

[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] G. Armitage, "Sensitivity of Quake3 Players To Network Latency," Poster session, SIGCOMM Internet Measurement Workshop, San Francisco, November 2001

[4] T. Henderson, "Latency and user behaviour on a multiplayer game server" Proceedings of the 3rd International Workshop on Networked Group Communications (NGC), London, UK, November 2001

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

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

[7] T. Beigbeder, R. Coughlan, C. Lusher, J.Plunkett, E. Agu, M. Claypool, "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003," ACM SIGCOMM 2004 workshop Netgames'04: Network and system support for games, Portland, USA, August 2004

[8] S.Zander, G.Armitage. "Empirically Measuring the QoS Sensitivity of Interactive Online Game Players," Australian Telecommunications Networks & Applications Conference 2004, (ATNAC2004), Sydney, Australia, December 8-10, 2004

[9] Chris Chambers, Wu-chang Feng, Wu-chi Feng, Debanjan Saha, "A Geographic, Redirection Service for On-line Games," in Proc. ACM Multimedia 2003 (short paper), November 2003

[10] "Wolfenstein Enemy Territory", http://games.activision.com/games/wolfenstein , (viewed on 6 August 2006)

[11] "GeoLite Country," http://www.maxmind.com/ app/geoip_country (viewed 6 August 2006)

[12] "Qstat Real-time Game Server Status", http://www.qstat.org (viewed 6 August 2006)

[13] G.Armitage, C.Javier, S.Zander, "Client RTT and Hop Count Distributions viewed from an Australian 'Enemy Territory' Server", CAIA Technical Report 060223A, 6th February, 2006 (http://caia.swin.edu.au/reports/060223A/CAIA-TR-060223A.pdf)

[14] P. Srisuresh, M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," RFC 2663, Internet Engineering Task Force, August 1999