Over-estimation of game server RTT during FPS server discovery

Grenville Armitage
Centre for Advanced Internet Architectures, Technical Report 080222B
Swinburne University of Technology
Melbourne, Australia
garmitage@swin.edu.au

***Abstract*—Online multiplayer first person shooter (FPS) games typically utilise a client-server architecture, and use a two-step process to find available game servers. Clients query a well-known master server for a list of registered game servers, then probe each listed game server in turn. This process emits thousands of probe packets over a short period of time. A client estimates the round trip time (RTT) to each server based on the time between emitting a probe and receiving a reply. Using examples from CounterStrike:Source this report illustrates how an excessive probe emission rate can artificially inflate the RTT to individual game servers.**

***Index Terms*—Server discovery, traffic optimisation**

I. Introduction

This report illustrates on how transmitting First Person Shooter (FPS) game server discovery probe packets too fast can inflate the RTT estimates reported to the player.

Internet-based multiplayer FPS games (such as Quake III Arena [1], Half-Life Counterstrike [2], and Half-Life 2 [3]) typically operate in a client-server mode, with game servers being hosted by Internet service providers (ISPs), dedicated game hosting companies and individual enthusiasts. Although individual FPS game servers typically only host from 4 to around 30+ players, there are usually many thousands of individually operated game servers active on the Internet at any given time. Due to the fast-pace and highly interactive nature of FPS games, players prefer game servers that exhibit lowround trip time (RTT, often colloquially known as 'lag'). Published literature suggests that competitive online FPS game play requires latencies below 150ms to 200ms [4]. This presents a challenge - how do game clients locate up-to-date information about all the servers available at any given time, such that the player can select a suitable server on which to play.

Server discovery operates similarly for many FPS games (due to the decentralised nature of game server hosting). First, a game client queries a master server unique to the particular game (pre-configured into the game client software). The master server returns a list of thousands (or tens of thousands) of IP addresses and port numbers representing game servers who have registered themselves as 'active'. The client then steps through this list, probing each listed game server for information (such as the current map type, game type and number of players). The probe is typically a brief UDP packet exchange, which allows the client to also estimate the RTT between itself and each game server. All this information is presented to the player (usually as it is gathered), who then selects a game server to join.

Server discovery is usually triggered explicitly by the human player running a particular game client. It may be triggered once or multiple times to refresh the list of available servers presented to the potential player by their client-side server browser. A given client will send out hundreds or thousands of probe packets to find and join only one game server. Consequently, individual game servers end up receiving, and responding to, tens of thousands of probe packets unrelated to the number of people actually playing (or likely to play) at any given time. The 'background noise' due to probe traffic fluctuates over time as game clients around the Internet startup and shutdown [5].

Many clients are connected to the Internet via 'broadband' connections in the 128Kbps - 2Mbps range (upstream). If probes are sent too slowly, it can take many minutes to probe thousands of servers. However, using examples from Valve Corporation's Counterstrike:Source (CS:S) [6], this report illustrates how an excessive probe emission rate can artificially inflate the RTT to individual game servers.

The rest of this report is organised as follows. Section II summarises the CS:S server discovery mechanism, section III describes the experimental results

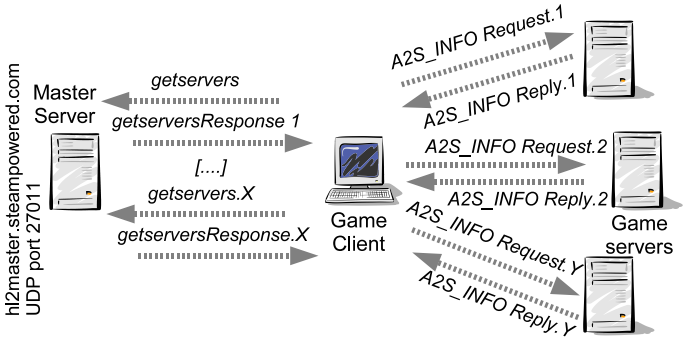

Figure 1. A Steam client discovering CS:S game servers

under three different probing scenarios, and the report concludes in section IV.

II. Valve's CS:S Server Discovery process

A. Server discovery protocol

CS:S was first released in late 2004 by Valve Corporation through their Steam online game delivery system. Public CS:S game servers register themselves with Steam's *master server* at hl2master.steampowered.com. Players initiate server discovery through their Steam client's game *server browser*. Figure 1 illustrates the key server discovery steps. (Note, Valve have no specific names for the messages between master server and client, so *getservers*, and *getserversResponse* have been chosen for clarity. UDP/IP packet formats for server discovery are available online [7].) As of late 2007, a Steam client would:

- Issue a *getservers* query to UDP port 27011 on the Steam master server
- Receive a *getserversResponse* packet containing the <IP address:port> pairs of up to 231 active game servers
- Send one *A2S_INFO Request* probe packet to each game server in order, eliciting an *A2S_INFO Reply* packet from every active game server
- Repeat the previous steps until the Steam master server has no more game server details to return

A2S_INFO Request UDP/IP packets are 53 bytes long and *A2S_INFO Reply* packets can average 135 bytes or longer.

A game server's estimated RTT is the time between the client sending an *A2S_INFO Request* and receiving the *A2S_INFO Reply*. Server-specific information in each *A2S_INFO Reply* is used to update the Steam client's on-screen server browser (along with estimated RTT) as replies come back. This process allows players to discover servers on which they might wish to play.

B. Client and master server filtering

Server-side filtering occurs when a player's initial *getservers* query requests game servers of a certain type (such as "only CS:S game servers") or servers believed (by the master server) to be in one of eight broad geographical regions of the planet (such as "US-West", "Europe", "Asia", etc). Server-side filtering can reduce the number of game servers returned by the master server, reducing the subsequent number of *A2S_INFO Request/Reply* probes and the time spent probing.

Client-side filtering (such as not showing servers that are full or empty, or ranking the servers in order of ascending RTT) simplifies the server browser's presentation of information. However, it occurs during or after each active probe and has limited impact on the traffic generated during server discovery.

C. Alternative server browsers

Other server browsers may use a different sequence. Qstat [8] (an open-source server browser) retrieves all registered servers first (using back to back *getserver* queries) before issuing *A2S_INFO Request* probes. Ultimately the same result - RTT to all active servers is estimated by probing them in the order their <IP address:port> pairs are returned by the master server.

D. Probing speed

The speed of server discovery is limited by a player's network connection. Players may configure their Steam client to assume a network connection of 'Modem - 56K', 'DSL > 256K', etc, thus influencing the *A2S_INFO Requests* transmission rate. In late 2007 and early 2008 the Steam master server was returning around 28-31K CS:S servers, of which ~27K respond to probes. At a nominal rate of 140 probes per second (approximating a Steam client configured for 'DSL > 256K' network access) CS:S server discovery takes ~230 seconds to complete. Alternative browsers (such as qstat) also provide mechanisms to adjust their probing rate.

However, too many probes per second can congest the player's link, inflating individual RTT estimates or causing probe packets to be dropped.

III. Experimentally observed probe traffic

A. Methodology

Two sources of probe traffic were used over a standard consumer-grade ADSL2+ connection based in Melbourne, Australia.

Qstat version 2.11 was used with default settings, and the Steam client was used at both low speed ("modem - 56K") and high speed ("DSL/Cable > 2M") settings. The Steam windows client showed the following under "Help":

- Built: Jan 9 2008, at 15:08:59
- Steam API: v007
- Steam package versions: 41 / 457.

The ADSL2+ link was a Belkin 4-port bridge and VoIP ATA, with the physical layer synchronised at 835 Kbps upstream and 10866 Kbps downstream. The link between home 100Mbps LAN and ISP was via PPPoE, bridged through the Belkin ADSL2+ modem (rather than terminating on the Belkin modem itself).

Both Qstat and Steam client ran on a host connected via 100Mbps ethernet to the ADSL2+ modem.

Each test involved initiating game server discovery and capturing the resulting *A2S_INFO Request* and *A2S_INFO Reply* packets in each direction. A number of tests were done using one of three scenarios (Qtstat default, Steam @ low speed and Steam @ high speed). Plots of estimated RTT versus time were constructed using the captured packet traces.

B. Results

Figures 2, 3 and 4 illustrate the estimated RTT to each game server as measured by Steam @ 35 probes/second (low speed), Qstat at 54 probes/second (default) and Steam @ 319 probes/second (high speed). Each dot represents an individual server being probed. (The average probe rates were calculated after the fact. Neither the Steam client nor Qstat allow probe rate to be stipulated in probes/second.)

The graphs clearly show that overall probe time reduces as the probe rate goes up, and that the general distribution of game servers is relatively consistent between tests. A small number of servers are within Australia or Asia (~20ms to ~140ms), a moderate number of servers are in North America (the ~200ms+ range) and a far larger community of game servers are in Europe (~330ms+). The need to jump oceans to reach Asia, North America and then Europe leads to a distinctly non-uniform distribution of RTTs.

Of most interest is Figure 4 showing a noticable 'smearing upwards' of estimated RTTs belonging to servers in different regions of the planet.

Recall that the minimum RTT is dictated by speed of light and geographical path between the client and server. The maximum RTT depends on the minimum RTT plus queuing delays along the path. In this experiment, the only thing being changed is the probe rate from the client. Figure 4 reveals that at 319 probes/second, a reasonable percentage of probes are incurring additional delay (relative to the RTTs seen in Figures 2and 3).

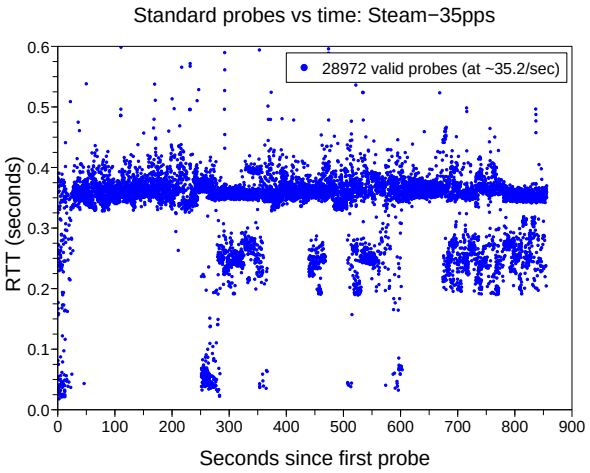

Figure 2. Estimated RTT versus time - Steam client at low speed ("modem - 56K") setting

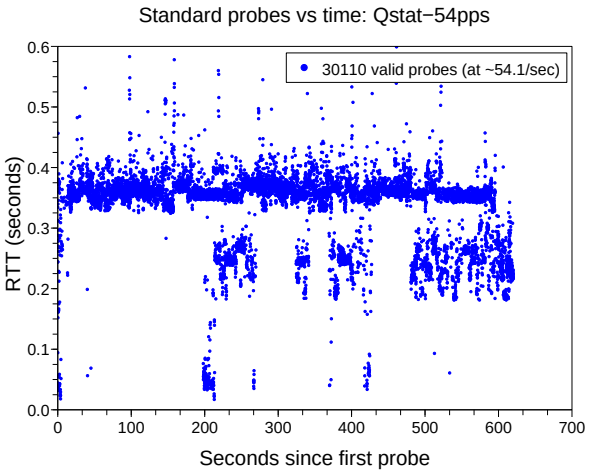

Figure 3. Estimated RTT versus time - Qstat client at default settings

The degree of RTT inflation can be more clearly seen in Figure 5 - a CDF of the RTT distributions of each test scenario. The Steam @ 319 probes/second curve is clearly inflating the RTTs of any servers in each geographical region. (Interestingly, the Steam @ 35 probes/second curve shows a slight increase in RTTs relative to the Qstat curve, even though Qstat was transmitting at a slightly higher number of probes/second.)

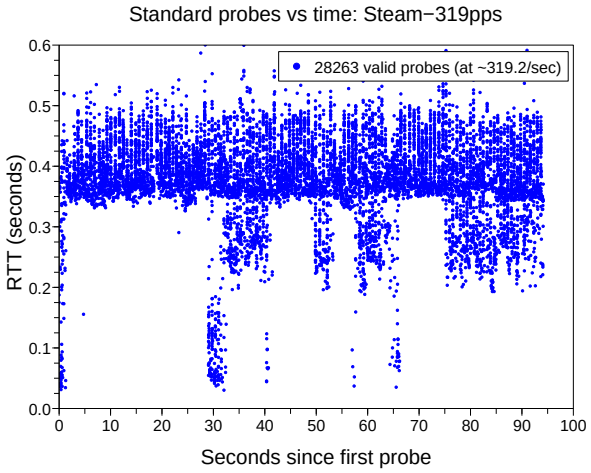

Figure 4. Estimated RTT versus time - Steam client at low speed ("DSL/Cable > 2M") setting

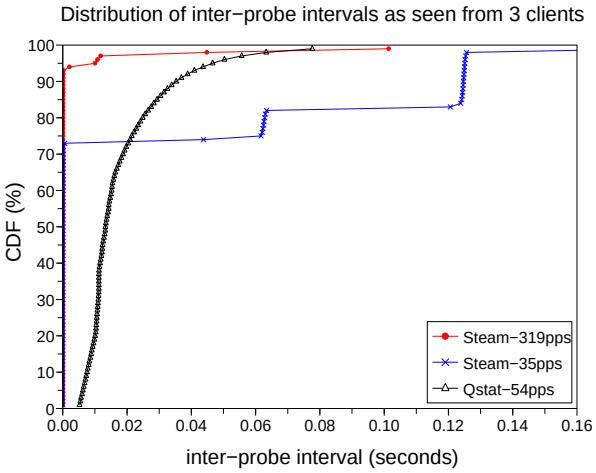

Figure 6. Distribution of inter-probe intervals from each test scenario

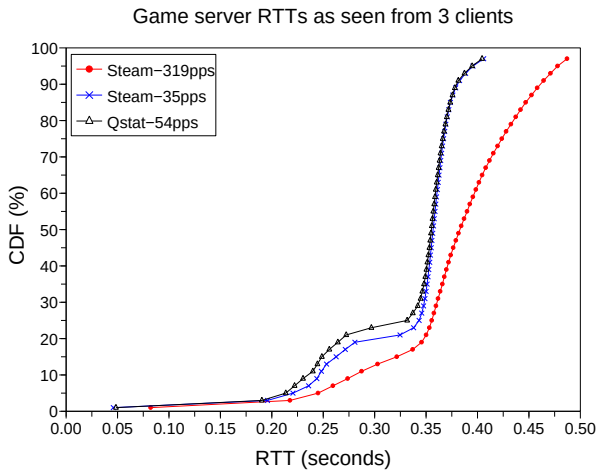

Figure 5. Distribution of estimated RTTs for each test scenario

C. Hypothesis

Queuing delay tends to occur when a sudden burst of packets converge on a network link and they cannot all be transmitted promptly. A queue holds excess packets waiting to be transmitted. A probe packet arriving when a queue is empty will experience shorter transit time than a packet arriving when a queue already has some packets in it. The size of the additional delay will depend on how many packets are already in the queue waiting for transmission.

Traffic between an online game client and remote servers will encounter packet queues in any locations - such as routers, switches, bridges and modems. This particular experiment is most likely triggering queuing in the consumer ADSL2+ modem. On one side of the modem is a 100Mbps home LAN, on the other side an asymmetric PPPoE link running one to two orders of magnitude slower (downstream and upstream respectively).

It is not sufficient that packets arrive over the modem's local LAN interface at an *average* rate below that of the modem's upstream ADSL2+ link. The pattern of individual packet arrivals is important. Groups of back-to-back probe packets cause more queuing than a smooth, regular stream of probe packets sent at short (yet regular) time intervals.

Figure 6 reveals that the Steam client is quite agressive in sending back-to-back bursts of outbound probe packets.

Each CDF reflects the distribution of time intervals between successive *A2S_INFO Request* packets during each test scenario. When running in "DSL/Cable > 2M" mode, over 90% of the probes emitted by the Steam client are less than 1ms apart. The resulting bursts of outbound probe packets would almost certainly create transient queues in the modem, adding to the measured RTT.

Note that it is not simply sufficiengt to 'slow down' the client. Even in "modem - 56K" mode (and one tenth the average probe rate) the Steam client emits 70% of its probe packets less than 1ms apart. In contrast, Qstat makes a modest attempt to 'pace' the transmission of probe packets, and thus the inter-probe intervals are spread over a relatively wide range. This might explain why Steam @ 35 probes/second in Figure 5 seems to have slightly higher RTTs on average than Qstat at 54 probes/second.

A good FPS client should attempt to smoothly pace

out the transmission of probe packets, in addition to ensuring the average probe packet rate does not exceed the player's available upstream network capacity.

IV. Conclusion

FPS game clients typically probe thousands of remote game servers during server discovery. Each probe is used to estimate the RTT to a single game server. By probing thousands of Counterstrike:Source game servers this report illustrates how an excessive (and bursty) probe emission rate can artificially inflate the estimated RTT to individual game servers. Consequently, game clients ought to smooth out their transmission of probe packets in order to minimise the RTTs reported during server discovery.

References

[1] id Software, *Quake III Arena*, http://www.idsoftware.com/, as of April 29th 2007.

[2] Valve Corporation, *CounterStrike: Source*, http://counter-strike.net/, accessed February 8th 2008.

[3] ——, *Half-Life 2*, http://half-life2.com/, as of April 29th 2007.

[4] G. Armitage, M. Claypool, and P. Branch, *Networking and Online Games - Understanding and Engineering Multiplayer Internet Games*. United Kingdom: John Wiley & Sons, Ltd., June 2006.

[5] S. Zander, D. Kennedy, and G. Armitage, "Dissecting server-discovery traffic patterns generated by multiplayer first person shooter games," in *Proceedings of ACM Networks and System Support for Games (NetGames) Workshop*, October 2005.

[6] Valve Corporation, *CounterStrike: Source*, http://counter-strike.net/, accessed February 8th 2008.

[7] ——, *Server Queries*, http://developer.valvesoftware.com/wiki/Server_Queries, as of February 7th 2008.

[8] *QStat*, http://www.qstat.org/, accessed February 8th 2008.