Measuring the auto-correlation of server to client traffic in First Person Shooter games

Philip Branch
Centre for Advanced Internet Architecture
Swinburne University of Technology
Melbourne, Australia
pbranch@swin.edu.au

Grenville Armitage
Centre for Advanced Internet Architecture
Swinburne University of Technology
Melbourne, Australia
garmitage@swin.edu.au

***Abstract*— Modelling traffic generated by Internet based multiplayer computer games has attracted a great deal of attention in the past few years. In part this has been driven by a desire to properly simulate the network impact of highly interactive online game genres such as the first person shooter (FPS). Past work has dealt with packet size and packet interarrival times without consideration of packet size autocorrelations. Packet size auto-correlation is an important element in the creation of plausible traffic generators for network simulators such as ns-2 and omnet++. In this paper we present an analysis of the auto-correlation of packet size for Half-Life, Half-Life Counterstrike, Half-Life 2, Half-Life 2 Counterstrike, Quake III Arena and Wolfenstein Enemy Territory. We show that packet sizes appear to be autocorrelated.**

Keywords- Network applications, games and services, Teletraffic Analysis, Traffic Engineering

I. INTRODUCTION

Modeling traffic generated by Internet based multiplayer computer games has attracted a great deal of attention in the past few years [2, 4-7, 10-12, 15]. Highly interactive genres such as the first person shooter (FPS) are of particular interest to network engineers. Like voice over IP (VoIP) and other interactive conference-style applications, FPS games are generally sensitive to packet loss, jitter and high latency. FPS games commonly use UDP rather than TCP, and transmission rates reflect in-game activity without any particular regard to network congestion. FPS games are typically based on a client-server model for network traffic, with thousands or tens of thousands of FPS servers active on the Internet at any given time [1]. This has motivated research community interest in predicting the traffic load imposed on network links by multiplayer FPS games. Since it is usually impractical to build and measure a full-size network, such analyses are usually investigated through simulation using statistical models created from the answers to the first question. Good traffic models are needed to ensure the simulations are useful [8].

Consequently, there has been a great deal of work in understanding traffic for the purpose of constructing simulation models of FPS games [1, 2, 5, 7, 10-12]. However, an important question so far neglected in these models relates to whether the sizes of successive packets are correlated or entirely random. If there is some form of correlation, is it positive where long packets tend to be followed by long packets and short packets tend to be followed by short packets, or is it negative where long packets tend to be followed by short packets and vice-versa? FPS traffic models developed so far have implicitly assumed that there is no correlation between packet lengths. Understanding correlation between packet lengths is important in the construction of realistic models. A succession of long packets followed by a succession of short packets will have a different impact on the network than a succession of long and short packets randomly interspersed. Long range traffic dependence has been a significant area of research for more than a decade [13] and has been shown to be present in many different forms of traffic. Consequently it seems reasonable to assume game traffic is also not entirely random.

Understanding correlation between packet lengths allows us to predict what happens to delay and delay variation when the traffic is multiplexed with other types of traffic and what link and server capacities are necessary to meet a given grade of service. In the same way that web and other traffic has been analyzed and modeled to predict the consequences for the Internet, it is necessary to analyze game traffic and produce models that can also be used in the same way [3].

Traffic in the client to server direction usually consists of small IP packets whose size distribution varies within a narrow band. On the other hand, traffic in the server to client direction consists of much larger packets that show great variation in size [1]. In this paper we investigate the autocorrelation of server to client packet lengths from FPS games with between 2 and 9 players. We use the public game traffic trace archives contained in Swinburne University of Technology's SONG database [14]. We investigate the autocorrelations from six FPS games released between 1998 and 2005: Half-Life Deathmatch (HLDM), Half-Life Counterstrike (HLCS), Quake III Arena (Q3A), Wolfenstein Enemy Territory (ET), Half-Life 2 Deathmatch (HL2DM) and Half-Life 2 Counterstrike (HL2CS).

We show that packet lengths appear to be auto-correlated for all six games and for any number of players. We show that simple Markov chain models may satisfactorily capture the correlation but that some games exhibit more complex, long-range correlations that may require more sophisticated traffic models.

The rest of our paper is structured as follows. Section II reviews the basic network architecture and traffic patterns of modern FPS games. Section III discusses Server to Client packet length autocorrelation. Section IV presents a number of autocorrelation plots from the six FPS games we tested. Section V shows how autocorrelation of a Markov process can be captured in a Markov chain transition matrix and Section VI concludes the paper.

II. First Person Shooter Games

In this section we review underlying reasons for the network traffic generated by modern FPS games. Readers familiar with FPS games may skip to section 3.

A. Client-server Architecture

Multiplayer online games have an underlying requirement that game-state information is shared amongst all players in something close to real-time. Each game client acts as an interface between the local human player and the virtual game-world within which the player interacts with other players. In principle clients might be designed to communicate directly with each other in a peer-to-peer fashion. In practice, most FPS games utilize a client-server model (including the six examples presented in this paper). Every client's actions are sent in short messages to the server, and every client is regularly updated with the actions taken by other players. The server implements the game-world's state machine, regulating client actions in order to maintain the game's internal rules and minimize opportunities for cheating.

B. Game-state Updates

A typical FPS game involves an ISP or game enthusiast hosting a game server on the Internet, and players joining the game using client software running on a home PC or IP-enabled game console. (In reality the game could also be run on a private, local IP network – commonly referred to as 'LAN parties'. For the purpose of this paper we focus on the case where both the game server and clients are all on the public Internet.) The game client updates and renders the game's virtual world on screen based on regular messages received from the game server. User inputs to the game client (actions such as walking, looking around or shooting weapons) are passed to the game server to be verified and propagated to other players.

Game-state updates must occur in a timely and prompt manner, with minimal bias or favor towards any particular player. Timeliness is achieved by sending a unicast IP packet to each client every Y milliseconds (ms). Y is typically in the range of 30 to 60ms – for example, the default update interval is 60ms for HLDM, 50ms for Q3A and 33ms for HL2DM. To minimize bias, update packets to different clients are sent in back-to-back bursts (for example, a four-player Q3A game server would default to sending bursts of four back-to-back update packets every 50ms, one IP packet to each active client). Each client will receive an update packet every Y ms regardless of how much in-game activity is occurring. The choice of Y for a given game depends on the available network capacity (longer Y for lower bandwidth demand) versus player expectations of smooth interactivity (shorter Y for more frequent updates).

Clients send their own updates to the game server at less precisely defined intervals, often influenced by the client's processor speed, graphics card settings and player activity. Typical intervals vary from 10ms to over 40ms [1].

C. Traffic Compression

Modern FPS games actively compress server to client data to maximize playability over a wide range of network conditions and consumer access technologies. Simple compression involves the use of smallest possible bit-fields to carry variable data. More complex compression involves the server only sending information to a client about regions of the virtual world currently visible to the client. Since every client has a different perspective on the virtual world the server effectively customizes every client update packet for the client to which it is sent.

Packets from server to client exhibit substantial variations in length as in-game activity surrounding a given client varies with time. For example, during active play of Q3A for a 9-player game, packets from server to client range between 32 and 960 bytes with 90% being between 98 and 460 bytes. For HL2DM packet lengths during active play are between 16 and 1400 bytes with 90% between 95 and 501 bytes.

A typical human can trigger only a limited number of events in any given 10ms to 40ms window. Consequently packets from client to server are typically much smaller than the packets from server to client, and exhibit very limited variation in size. For example, client to server IP payload lengths range between 25 and 45 bytes for Q3A during active game play, with 90% of all packets between 28 and 38 bytes long. For HL2DM, packet lengths vary between 36 and 99 with 90% of all packets being between 57 and 75 bytes long.

D. Phases of game-play and game traffic

There are roughly three different phases of interaction between client and server that impact on network traffic.

- A client initially connects to the server, and receives data from the server to update the client's local virtual world information (map definitions, avatar 'skins', etc)
- The client is connected to the server and game is in progress (players running around shooting and interacting with each other)
- The client is connected to the server, and the game has been paused as the server changes maps or restarts a previous map (after someone wins the previous 'round')

Tight control over network jitter and packet loss is really only required during active game-play. During periods of player inactivity (initial client connection and server changing maps) the network can exhibit fluctuating latency, jitter and packet loss without upsetting the player. Our analysis is of data obtained during active game-play.

III. SERVER TO CLIENT PACKET LENGTH AUTOCORRELATION

In this section we discuss the need for empirical experiments to model basic server to client packet traffic, and introduce the autocorrelation plot as a way of measuring correlation between successive packet lengths.

A. Modeling server to client traffic from empirical data

A primary motive for modeling FPS network traffic is to assist in provisioning the network to provide a quality game playing experience. Therefore we simplify the modeling process by focusing on traffic patterns that exist during active game play. We further focus on server to client packet size distributions, as client to server packet length distributions are usually quite simple.

In principle one could estimate the correlation of server to client packet sizes by applying models of player mobility and behavior onto a given map. Simulated interactions would lead to simulated in-game events and hence simulated server to client packets. However, such an approach is likely to be rather complex, and begs the question of where to find realistic player mobility models applicable to each map.

In practice it is easier to simply gather server to client packet statistics while observing actual games in progress. Trials can be run and monitored for specific maps, and specific numbers of players. The resulting server to client packet size distributions will reflect each player's natural movements and the player-player interactions induced by the particular map.

B. Measuring packet length autocorrelation

A purely random sequence of packet lengths will exhibit near zero correlation with successive packet lengths whereas if the traffic is not random the correlation between successive packet lengths will be significantly greater than zero.

Autocorrelation plots are commonly used for investigating the randomness or otherwise of a data set. An autocorrelation plot shows the autocorrelations between the data for varying packet shifts (often referred to as 'lag').

The simplest autocorrelation models (apart from purely random) are Markov models. In Markov models prediction of the next value (in this case packet length) is based solely on the current value. An indication of whether or not traffic exhibits Markov characteristics is the rate at which correlation between two samples decreases as the distance between the samples increases [13]. For a Markov process, the correlation between two samples decreases exponentially as the distance (or lag) time between the samples increases. For a Process showing longer range dependence the correlation decreases more slowly.

IV. AUTOCORRELATION PLOTS

In this section we show autocorrelation plots for HLDM, HLCS, Q3A, ET, HL2DM, and HL2CS. The plots show autocorrelation spanning 250 packets, corresponding to approximately 5 seconds of game play. For reasons of space we show only 2- and 9- player games. Each plot contains both the empirically derived autocorrelation function and a best-fit exponential function. If the autocorrelation function decays more quickly than the exponential function then that is some evidence of randomness. If the autocorrelation function is approximately exponential, then that is evidence that the packet length autocorrelation can be captured with a Markov model. If the autocorrelation function decays much more slowly than the exponential function then that is evidence of a more complex, long rang dependence.

All plots suggest that the games exhibit autocorrelation. Whether or not Markov models are adequate will depend on the purpose of the simulation but from the plots it appears that Q3A, ET, HL2DM and HL2CS can be adequately modeled by Markov methods. HLDM and HLCS appear to exhibit longer-range dependence that may require more complex models.

A. Half-Life

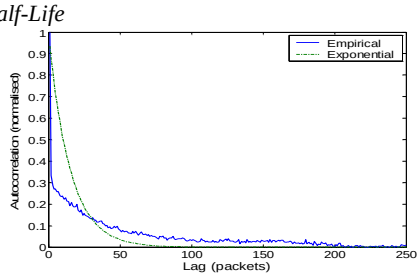

Figure 1. Half-Life Deathmatch empirical and exponential 2-player autocorrelation functions

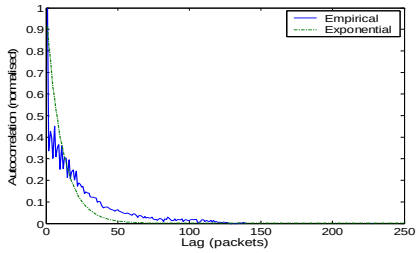

Figure 2. Half-Life Deathmatch empirical and exponential 9-player autocorrelation functions

B. Half-Life Counterstrike

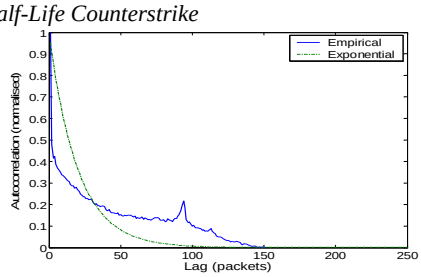

Figure 3. Half-Life Counterstrike empirical and exponential 2-player autocorrelation functions

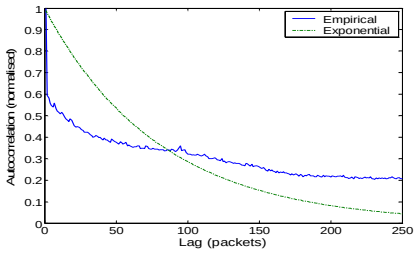

Figure 4. Half-Life Counterstrike empirical and exponential 9-player autocorrelation functions

C. Quake III Arena

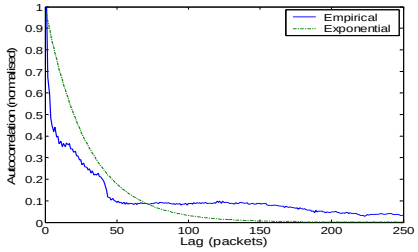

Figure 5. Quake III Arena empirical and exponential 2-player autocorrelation functions

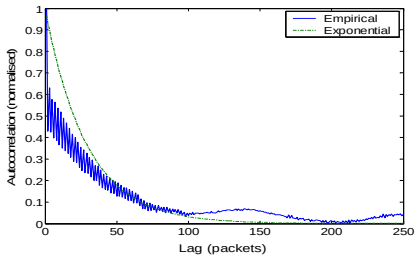

Figure 6. Quake III Arena empirical and exponential 9-player autocorrelation functions

D. Wolfenstein Enemy Territory

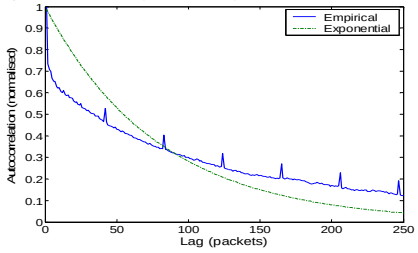

Figure 7. Wolfenstein Enemy Territory empirical and exponential 2-player autocorrelation functions

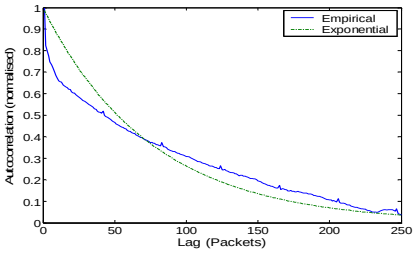

Figure 8. Wolfenstein Enemy Territory empirical and exponential 9-player autocorrelation functions

E. Half-Life 2

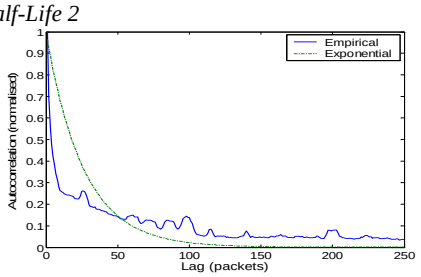

Figure 9. Half-Life 2 Deathmatch empirical and exponential 2-player autocorrelation functions

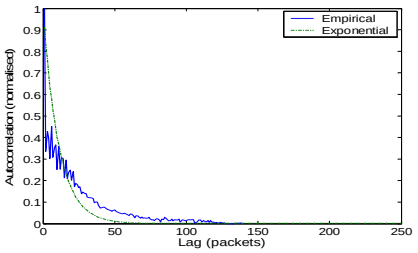

Figure 10. Half-Life 2 Deathmatch empirical and exponential 9-player autocorrelation functions

F. Half-Life 2 Counterstrike

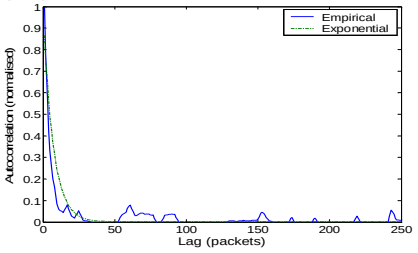

Figure 11. Half-Life 2 Counterstrike empirical and exponential 2-player autocorrelation functions

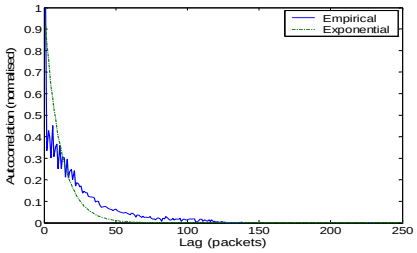

Figure 12. Half-Life 2 Counterstrike empirical and exponential 9-player autocorrelation functions

V. Markov Chain Representation of Autocorrelation

In the previous section we have shown that there is evidence of autocorrelation in packet lengths generated by an FPS game. We now give a brief outline of how such autocorrelation can be implemented in a simulation. We will illustrate this using the HL2CS 9-player game whose autocorrelation function is shown in Figure 12. From the autocorrelation function we can see that the autocorrelation function is approximately exponential and so can reasonably be modeled with a Markov process. Because it is easy to implement in a simulation we will illustrate how this data can be implemented with a discrete time Markov Chain [9].

A Markov chain is a sequence of random variables, X_1, X_2, X_3,… with the property that the future state is dependent only on the current state. That is:

$$\Pr(X_{n+1} = x_j \mid Xn = x_i, X_{n-1} = x_k, \ldots) = \Pr(X_{n+1} = x_j \mid Xn = x_i)$$

Where the number of states x_i is finite, the conditional probabilities can be represented by a transition matrix **T**. If the process is in state **i**, the probability of the next state being state **j** is given by the element $\mathbf{T}_{i,j}$

From the data used to construct the autocorrelation plot for HL2CS, we can construct the transition matrix **T** of conditional probabilities. The transition matrix for the 9-player Half-Life 2 Counterstrike game is shown in the following matrix.

$$\mathbf{T} = \begin{bmatrix} 0.82 & 0.17 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.04 & 0.85 & 0.09 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.35 & 0.57 & 0.06 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.01 & 0.22 & 0.43 & 0.29 & 0.03 & 0.01 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.18 & 0.38 & 0.16 & 0.20 & 0.05 & 0.01 & 0.00 & 0.00 \\ 0.06 & 0.16 & 0.19 & 0.10 & 0.11 & 0.34 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.16 & 0.25 & 0.06 & 0.05 & 0.09 & 0.35 & 0.03 & 0.00 \\ 0.00 & 0.13 & 0.11 & 0.18 & 0.09 & 0.20 & 0.07 & 0.16 & 0.07 \\ 0.00 & 0.15 & 0.08 & 0.08 & 0.08 & 0.19 & 0.27 & 0.08 & 0.08 \end{bmatrix}$$

In this matrix we have 'binned' packet lengths into 0 to 50 bytes, 51 to 100, 101 to 150 and so on. The matrix can be used to generate correlated output by going from bin **i** to bin **j** with probability $\mathbf{T}_{i,j}$ and randomly choosing a packet length in the range represented by that bin. By binning the data we avoid over-fitting the data set and keep the transition matrix to a manageable size.

VI. Conclusion

Previous work modeling server to client FPS traffic has typically made a simplifying implicit assumption that packet lengths are uncorrelated. We have shown this assumption to be of doubtful validity. Empirical evidence from six modern FPS games, with varying numbers of players, reveals packet length autocorrelation. We have shown how this autocorrelation can be captured in a Markov Chain model.

Future work will involve investigating the effectiveness or otherwise of Markov process models in capturing the nature of this autocorrelation.

Acknowledgment

This work was partly supported by the Smart Internet Cooperative Research Centre http://www.smartinternet.com.au

References

[1] Armitage, G., Claypoole, M. and Branch, P., "*Networking and Online Games : Understanding and Engineering Multiplayer Internet Games*," John Wiley and Sons Ltd, Chichester, England, 2006.

[2] Borella, M., "Source models of network game traffic," *Computer Communications*, *23* (4). 403-410.

[3] Cunha, C., Bestavros, A. and Crovella, M., "Characteristics of WWW Client-based Traces," *Boston University Technical Report*, 19951995

[4] Farber, J., "Network game traffic modelling," in *Proc of the first ACM workshop on network and system support for games*, (Braunschweig, Germany, April 2002).

[5] Farber, J., "Traffic Modelling for Fast Action Network Games," *Multimedia Tools and Applications*, *23* (1). 31-46.

[6] Feng, W., Chang, F., Feng, W. and Walpole, J., "Provisioning on-line games: a traffic analysis of a busy Counter-Strike server," in *Proc. of SIGCOMM Internet Measurement Workshop*, (Marseille, France, November 2002).

[7] Feng, W.-C., Chang, F., Feng, W.-C. and Walpole, J., "A traffic characterization of popular on-line games," *IEEE/ACM Transactions on Networking*, *13* (3).

[8] Floyd, S. and Kohler, E., "Internet Research Needs Better Models," in *First Workshop on Hot Topics in Networks*, (Princeton, New Jersey, 28-29 October).

[9] Kemeny, J., Knapp, A. and Snell, J., "*Denumerable Markov Chains*," von-Nostrand, Princetone, N. J., 1976.

[10] Lang, T. and Armitage, G., "A ns2 model for the Xbox system link game HALO," in *Proc. Australian Telecommunications Networks and Applications Conference*, (Melbourne, Australia, December 2003).

[11] Lang, T., Armitage, G., Branch, P. and Choo, H., "A synthetic traffic model for Half-Life," in *Proc. of the Australian Telecommunications Network and Applications Conference*, (Melbourne, December 2003).

[12] Lang, T., Branch, P. and Armitage, G., "A synthetic model for Quake III traffic," in *Proc. ACM SIGCHI Advances in Computer Entertainment (ACE2004)*, (Singapore, June 2004).

[13] Paxson, V., "Empirically derived analytic models of wide-area TCP connections," *IEEE/ACM Transactions on Networking*, *2* (4). 316-336.

[14] Swinburne University of Technology, "Simulating Online Network Games (SONG) database," 2006 http://caia.swin.edu.au/sitcrc, 27 July2006

[15] Zander, S. and Armitage, G., "A traffic model for the XBOX game Halo 2," in *15th ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2005)*, (Washington, June 2005).