Extrapolating Server To Client IP traffic From Empirical Measurements of First Person Shooter games

Philip Branch
Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology
Melbourne Victoria, Australia
(+613) 9214 8000
pbranch@swin.edu.au

Grenville Armitage
Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology
Melbourne Victoria, Australia
(+613) 9214 8000
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). Packet size distributions are an important element in the creation of plausible traffic generators for network simulators such as ns-2 and omnet++. In this paper we present a simple technique for creating representative packet size distributions for N-player FPS games based on empirically measured traffic of 2- and 3-player games. We illustrate the likely generality of our approach using data from Half-Life, Half-Life Counterstrike, Half-Life 2, Half-Life 2 Counterstrike, Quake III Arena and Wolfenstein Enemy Territory.

Categories and Subject Descriptors
C.2.3 [Computer-Communications Networks]: Network Operations–network monitoring; C.2.5 [Computer-Communications Networks]: Local and Wide-Area Networks–Internet

General Terms
Measurement, Performance, Design.

Keywords
First Person Shooter, Games, Teletraffic Analysis, Traffic Engineering

1. 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, 9-11, 14]. 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 intolerant of packet loss, jitter and high latency. FPS games commonly use UDP for transport and adjust packet transmission rates based solely on in-game activity rather than network congestion. Finally, 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 and simulating the traffic load imposed on network links by multiplayer FPS games.

Netgames'06, October 30–31, 2006, Singapore.

Two questions are of particular interest - how traffic generated by FPS games increases as the number of players increases, and how this traffic affects, and is affected by, other traffic sharing the network. Since it is usually impractical to build and measure a full-size network, the second question is typically answered 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].

Understanding how game traffic varies as the number of users increases 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, and the models used 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 is independent of the number of players on a given server. On the other hand, traffic in the server to client direction usually shows distinct variation as the number of players increases [1] Published work to date has typically involved empirical studies of FPS games in small test beds with up to 8 to 10 players. Traffic models are created that match the statistical packet size distributions for each N-player game, for N = 2, 3, and so on. Yet many public FPS game servers are configured to allow up to 20, 30 or over 40 players each.

In this paper we propose and illustrate a technique for extrapolating the N-player traffic statistics from empirically measured traffic of 2- and 3-player FPS games. In particular we focus on predicting the distribution of packet size in the server to client direction. This allows small-scale empirical measurements (for example, from public game traffic trace archives such as Swinburne University of Technology's SONG database [13]) to be applied to larger scale simulations of FPS traffic acting on an IP network. We illustrate the likely

generality of our approach using data from six FPS games released between 1998 and 2005: Half-Life, Half-Life Counterstrike, Quake III Arena, Wolfenstein Enemy Territory, Half-Life 2 and Half-Life 2 Counterstrike.

The rest of our paper is structured as follows. Section 2 reviews the basic network architecture and traffic patterns of modern FPS games. Section 3 discusses our decision to model server to client packet size distributions and introduces our proposed method of predicting N-player distributions. Section 4 introduces a number of example distributions from the six FPS games we tested. Section 5 assesses the predictive capability of our proposed scheme when applied to a wide range of empirically derived scenarios. Section 6 discusses future research, and section 7 concludes the paper.

2. 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.

2.1 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.

2.2 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 Half Life Deathmatch, 50ms for Quake III Arena and 33ms for Half-Life 2 Deathmatch. To minimize bias, update packets to different clients are sent in back-to-back bursts (for example, a four-player Quake III Arena 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].

2.3 Traffic compression

To maximize playability over a wide range of network conditions and consumer access technologies modern FPS games actively compress the data sent over the network. 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. (Although this minimizes the size of server to client packets, it also reduces the potential utility of multicasting server update packets to every client.)

Clients generate events describing a single player's activity. 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 Quake III Arena during active game play, with 90% of all packets between 28 and 38 bytes long. For Half-Life 2 Deathmatch, packet lengths vary between 36 and 99 with 90% of all packets being between 57 and 75 bytes long.

Packets in the server to client direction exhibit substantial variations in length as in-game activity surrounding a given client varies with time. For example, during active play of Quake III Arena 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 Half-Life 2 Deathmatch packet lengths during active play are between 16 and 1400 bytes with 90% between 95 and 501 bytes.

The in-game activity conveyed in a single update packet includes a component proportional to the number of other players visible to a client at a point in time. The actual visibility of other players, and what they are doing at the time, itself depends on the number of players and the virtual world's layout (the 'map'). For example, maps with many walls and corridors will result in less visibility between players (and less information per update packet on average) than maps with wide-open areas. Likewise, a map containing many players will experience many more player-player interactions (per unit time) than a map with few players scattered around.

2.4 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.

3. MODELING SERVER TO CLIENT TRAFFIC

In this section we discuss the need for empirical experiments to model basic server to client packet traffic, and introduce our technique for extrapolating these statistics to cover N-player games.

3.1 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 distribution 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.

3.2 Extrapolating from 2- and 3-player games

Building reasonable statistical models requires controlled, repeated empirical studies – and these require dedicated hardware and multiple committed players. Not surprisingly most controlled trials end up using less than 10 players. However, FPS game servers may well support between 16 and 60+ players (subject to network capacity at the server and the server's own CPU capacity). We propose a technique whereby controlled trials with small numbers of players can be used to synthesize server to client packet distributions for larger numbers of players on the same map.

If we make a number of simplifying assumptions we can develop a technique for predicting the Probability Mass Function (PMF) of packet length for larger numbers of players given the PMF of small player games. The PMF function is similar to the Probability Density Function except that the PMF is used to describe random variables that take on discrete values. Within the game teletraffic literature the terms Probability Density Function (PDF) and Probability Mass Function (PMF) tend to be used interchangeably. However, since we are dealing with packet lengths, which can only take on discrete values, Probability Mass Function is the term we will use. In this and the following section we illustrate how successful predictions as to the PMF of four to nine player games can be made if the PMF of a two and three player game is known.

The PMF of a two-player game gives information about the behavior of each player in isolation and of his or her interactions with the other player. In showing how the PMF of larger numbers of players can be predicted from games played by smaller numbers of players we make the following simplifying assumptions:

- The nature of game play for individual players does not significantly change regardless of the number of players. Each player spends similar amounts of time involved in exploring the map, collecting useful items and engaging in battles regardless of the number of players.
- Players have similar behavior. They may not be of similar ability but will engage in similar activities in much the same way as each other.

Essentially these assumptions propose that the random variable describing the packet length of an N-player game can be constructed through adding together the random variables of smaller player games. Because we assume that individual game play does not change as the number of players increases and that the game players have similar behavior, we propose that, for example, the random variable describing the length of packet payloads generated by a 5-player game can be constructed from adding the two random variables that describe a 2-player game and a 3-player game. Of course these are simplifying assumptions that only approximate the true nature of FPS games. Nevertheless, by making them we can develop a simple technique for predicting the PMF of games with larger number of players.

We now formalize this analysis. Each payload is made up of a fixed component describing static details of the game and a variable component describing the dynamic details (based on player behavior) of the game. The fixed component is unchanged regardless of the number of players in the game.

If we denote the variable component of the payload of an N-player game by X_N and the PMF by f_{XN} then our assumptions enable us to predict that the X_4 will equal $X_2 + X_2$, that X_5 will equal $X_2 + X_3$ and so on.

Since the PMF of the sum of two or more random variables is the convolution of their PMFs, the PMF of a 4-player game f_{X4}

will be $f_{X2} * f_{X2}$ (where * denotes convolution), the PMF of a 5-player game will be given by $f_{X2} * f_{X3}$ and so on.

When developing a simulator of game traffic, knowing what happens to the distribution of packet length as the number of players increases is important. Packet length distributions have consequences for network, server and client capacity planning.

In the next two sections we illustrate the use of this approach to synthesize N-player packet size distributions using empirically derived 2- and 3-player server to client distributions.

4. SERVER TO CLIENT PACKET LENGTH DISTRIBUTIONS

In this section we use publicly available traffic data taken from [13] to illustrate the approach described above.

4.1 Packet size versus number of players

First we take two examples, Quake III Arena (Figure 1) and Half-Life 2 (Figure 2) to illustrate how the spread of the PMFs of packet length increases in a reasonably smooth fashion as a function of the number of players. (When more players are active on the server, more information is, on average, conveyed to each individual player in each server to client update packet.)

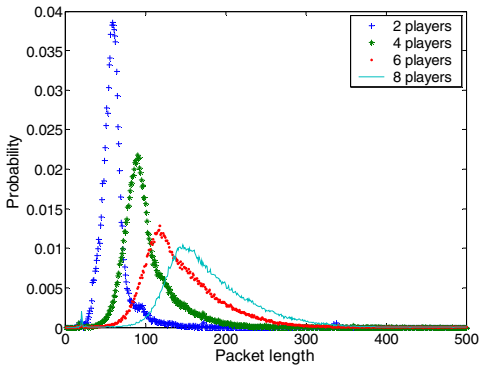

Figure 1. Quake III Arena Packet Length PMF for 2 to 8 players

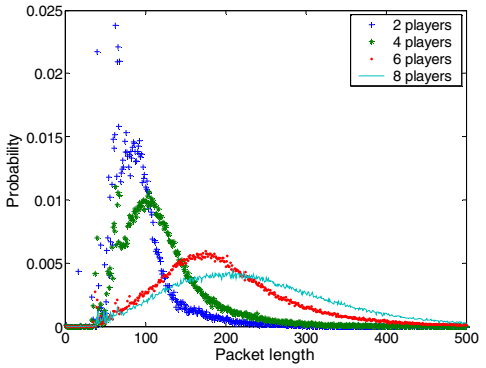

Figure 2. Half-Life 2 Packet Length PMF for 2 to 8 players

A direct consequence of the simplifying assumptions in section 3 is that the average packet length should have a linear relationship with the number of players. If $X_4 = X_2 + X_2$, and $X_6 = X_2 + X_2 + X_2$ then the mean of X_4 will be $|X_4| = 2|X_2|$ and $|X_6| = 3|X_2|$ and so on.

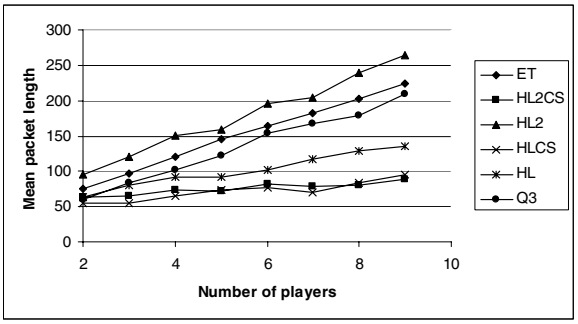

Figure 3. Mean packet size vs. number of players

Using all six FPS games studied in this paper, Figure 3 shows that the mean of the empirically derived packet length distributions does increase in an approximately linear fashion between 3 and 9 players. The means are shown for Half-Life (HLDM), Quake III Arena (Q3), Half-Life 2 (HL2DM), Half-Life Counterstrike (HLCS), Wolfenstein Enemy Territory (ET) and Half-Life 2 Counterstrike (HL2CS). This suggests that our simplifying assumptions in section 3 are not unreasonable. We now show how to use the assumptions to synthesize the server to client packet size PMF of games with larger numbers of players.

4.2 Synthesizing an N-player packet size distribution

Our model predicts that the random variable describing the variable part of the distribution for each game can be constructed by summing different combinations of X_2 and X_3 random variables. In this section we illustrate the process for Quake III Arena and Half-Life 2 by predicting f_{X4} and f_{X8} with suitable convolutions of f_{X2} and f_{X3}.

It is important to note that we make no assumptions as to the distribution of the X_2 and X_3 random variables. We use these random variables as building blocks to predict the PMF of games with larger numbers of players.

To obtain the synthetic PMFs we went through the following steps:

1. Obtain statistics of traffic during active game play. The startup phase of most games involves the exchange of many small 'keep-alive' packets while all players join the game. A sufficiently large number of statistics need to be captured. For the 2- and 3-player games we captured active game play statistics for at least twenty minutes, corresponding to approximately 48,000 packets [13]. In our experiments we captured data using tcpdump.
2. Extract the PMF from the statistics. We used MATLAB for this step.

3. Determine the length of the fixed component of each packet by inspecting the PMF. (For Quake III the base length is 32, for Half Life 2 the base length is 40.)
4. Carry out the necessary convolutions to predict the PMF of packet length of games with larger number of players. We used MATLAB for this step as well.
5. A consequence of the convolution will be that for each convolution the PMF will have been shifted right by the fixed component length of the packet. Since this is a fixed value regardless of the number of players, the PMF needs to be shifted left by the same amount. (An alternative is to use only the variable component of the packet length in the convolutions. However, we found the approach described here easier to implement.)

There is often more than one way in which a synthesized PMF can be determined from f_{X2} and f_{X3} . For example f_{X6} could be determined from $f_{X3}*f_{X3}$ or from $f_{X2}*f_{X2}*f_{X2}$. In our modeling we synthesized the packet length PMF for 4- to 9-player games in the following way:

$$f_{X4} = f_{X2}*f_{X2}$$
$$f_{X5} = f_{X3}*f_{X2}$$
$$f_{X6} = f_{X3}*f_{X3}$$
$$f_{X7} = f_{X2}*f_{X2}*f_{X3}$$
$$f_{X8} = f_{X2}*f_{X2}*f_{X2}*f_{X2}$$
$$f_{X9} = f_{X2}*f_{X2}*f_{X2}*f_{X3}$$

We chose this combination in order to minimize the number of convolutions necessary to predict the PMF of the larger player game using only the 2 and 3 player games.

Figure 4 and Figure 5 show the empirical and synthetic PMFs for Quake III Arena and Half-Life 2 6-player games. These were obtained as a convolution of two 3-player games. The agreement between the empirical and synthetic models is very good.

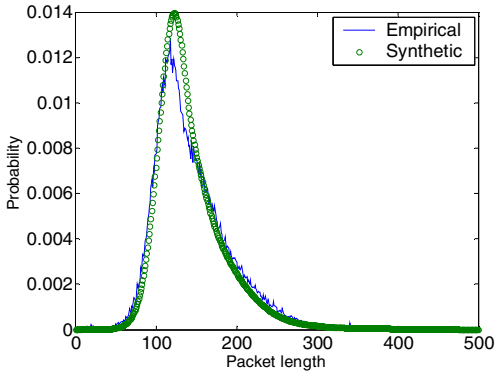

Figure 4. Quake III Arena empirical and synthetic PMF for 6-player game

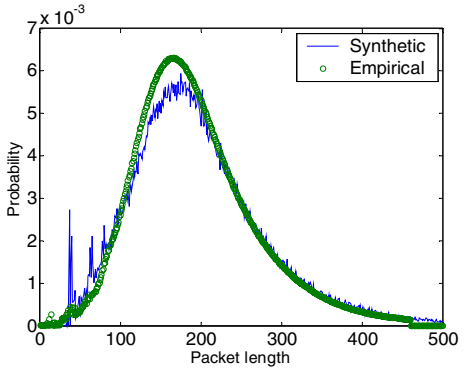

Figure 5. Half Life 2 empirical and synthetic PMF for 6-player game

Figure 6 and Figure 7 show the synthetic and empirical PMFs for Quake III and Half-Life 2 8-player games. These were obtained as a convolution of four 2-player games. Once again, the agreement between the empirical and synthetic models is very good.

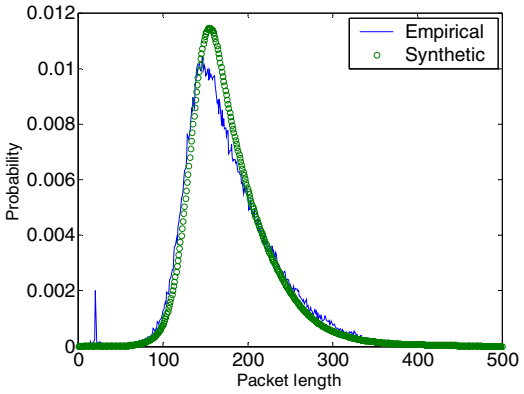

Figure 6. Quake III Arena empirical and synthetic PMF for 8-player game

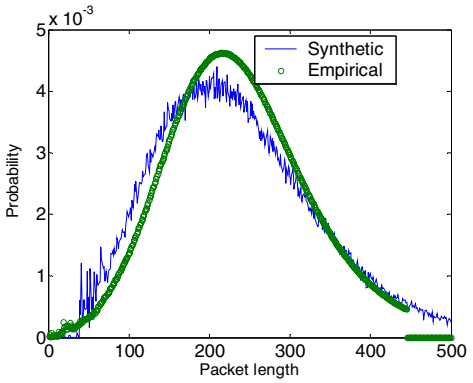

Figure 7. Half-Life 2 empirical and synthetic PMF for 8-player game

5. EXTRAPOLATING TO N-PLAYER GAMES

5.1 Comparison of empirical and estimated distributions

In this section we present plots showing synthetic and empirically derived probability distributions for the six FPS games. Half-Life (HLDM), Quake III Arena (Q3), and Half-Life 2 (HL2DM) traces were gathered in 'deathmatch' mode, where all players compete against each other. Half-Life Counterstrike (HLCS), Wolfenstein Enemy Territory (ET) and Half-Life 2 Counterstrike (HL2CS) traces were gathered in team-play mode, where players (having different character roles) are aligned with one of two teams and battle the other team's members [1]. The empirical data is taken from [13].

Due to space constraints we only show a representative sample of PMF plots along with a Q-Q plot for each game. Q-Q plots are used to obtain a qualitative indication of how well two distributions (in this case an empirical and synthetic distribution) match each other. They are obtained by plotting the quantiles of the two cumulative distributions against each other. The nearer to a straight line between (0,0) and (1,1) the better the match between the two distributions.

5.1.1 Half-Life

Half-Life shows good agreement between the empirical and synthetic models in Figure 8, Figure 9 and Figure 10.

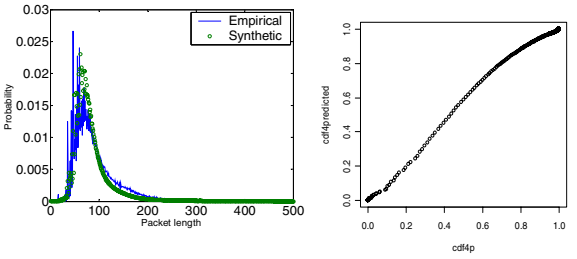

Figure 8. Four player game PMF and Q-Q plot (HLDM)

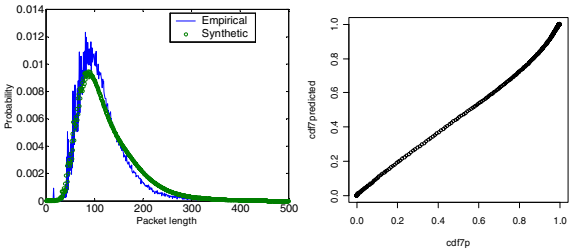

Figure 9. Seven player game PMF and Q-Q plot (HLDM)

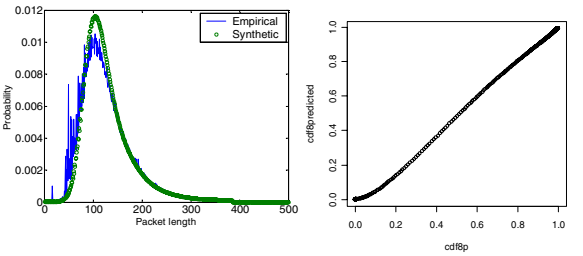

Figure 10. Eight player game PMF and Q-Q plot (HLDM)

5.1.2 Half-Life Counterstrike

Being a team game, we might expect Half-Life Counterstrike to deviate somewhat from our assumption of each player having similar behavior. Nevertheless, we still see good agreement between the empirical and predicted behavior in Figure 11, Figure 12 and Figure 13. (The eight-player game does exhibit some divergence revealed as small peaks at smaller packet lengths in the PMF plots. Yet even so, the Q-Q plot shows a good match).

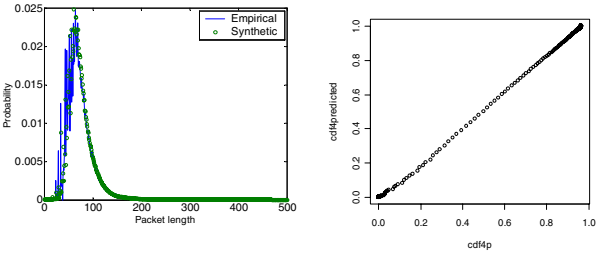

Figure 11. Four player game PMF and Q-Q plot (HLCS)

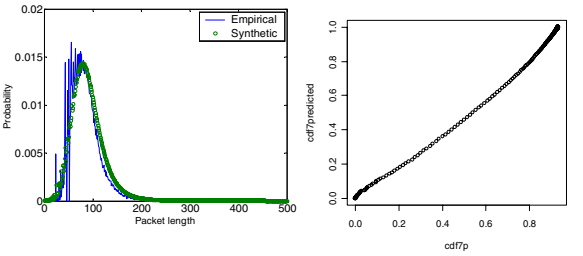

Figure 12. Seven player game PMF and Q-Q plot (HLCS)

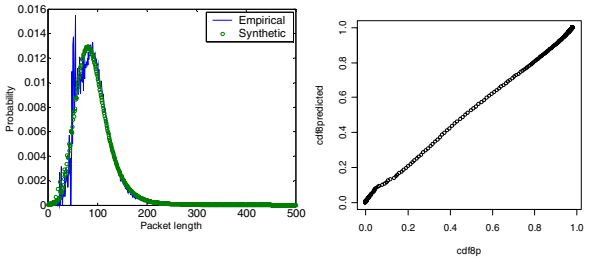

Figure 13. Eight player game PMF and Q-Q plot (HLCS)

5.1.3 Quake III Arena

Quake III Arena also demonstrates good agreement between the synthetic and empirical models in Figure 14, Figure 15 and Figure 16.

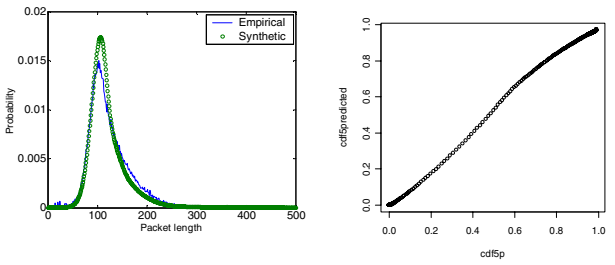

Figure 14. Five player game PMF and Q-Q plot (Q3)

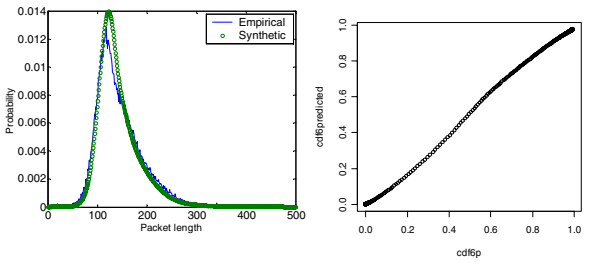

Figure 15. Six player game PMF and Q-Q plot (Q3)

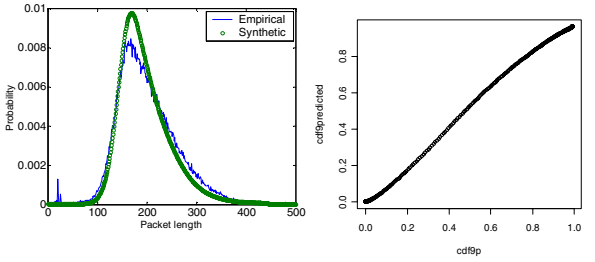

Figure 16. Nine player game PMF and Q-Q plot (Q3)

5.1.4 Wolfenstein: Enemy Territory (ET)

Being a team game, we might expect Wolfenstein: Enemy Territory to deviate somewhat from our assumption of each player having similar behavior. Nevertheless, we still see good agreement between the empirical and predicted behavior in Figure 17, Figure 18 and Figure 19. (The eight-player game does exhibit some divergence revealed as small peaks at smaller packet lengths in the PMF plots. Yet even so, the Q-Q plot shows a good match).

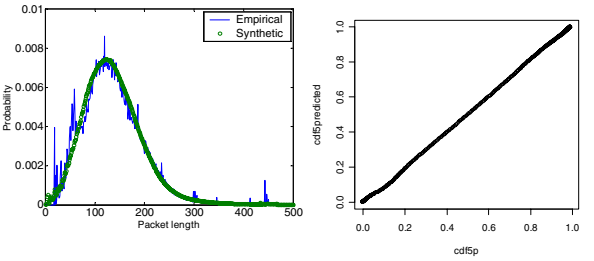

Figure 17. Five player game PMF and Q-Q plot (ET)

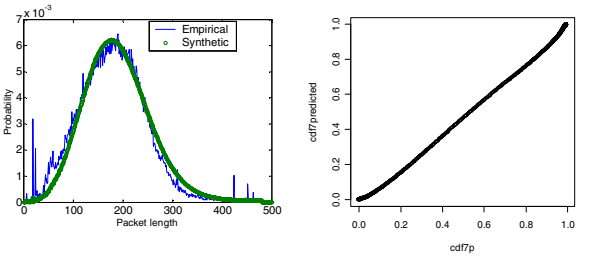

Figure 18. Seven player game PMF and Q-Q plot (ET)

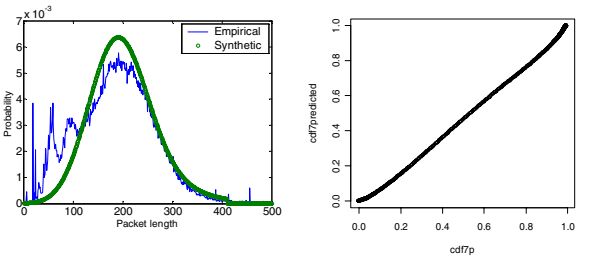

Figure 19. Eight player game PMF and Q-Q plot (ET)

5.1.5 Half-Life 2

Half-Life 2 reveals good agreement between the synthetic and empirical models as shown in Figure 20, Figure 21 and Figure 22.

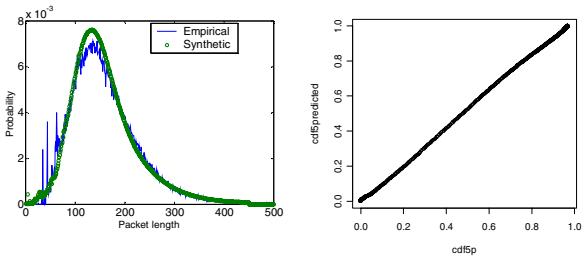

Figure 20. Five player game PMF and Q-Q plot (HL2DM)

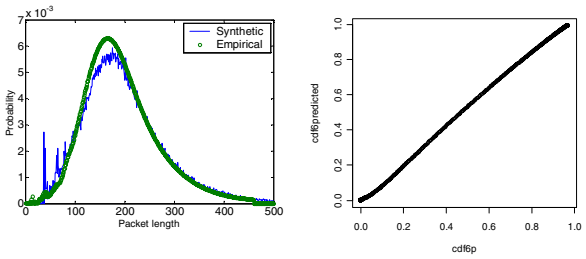

Figure 21. Six player game PMF and Q-Q plot (HL2DM)

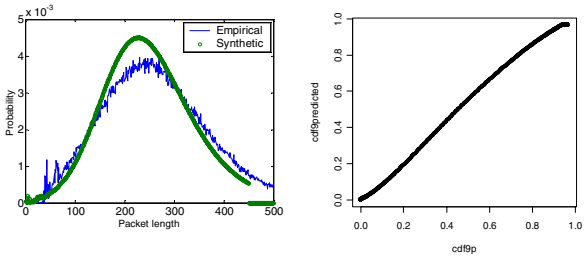

Figure 22. Nine player game PMF and Q-Q plot (HL2DM)

5.1.6 Half-Life 2 Counterstrike

Half-Life 2 Counterstrike also shows good agreement between the empirical and predicted distributions in Figure 23, Figure 24, and Figure 25. However, as with the other team games, there are some peaks among the smaller packet lengths that are difficult to capture in the synthetic model. Nevertheless, the Q-Q plots show the empirical and synthetic models match reasonably well.

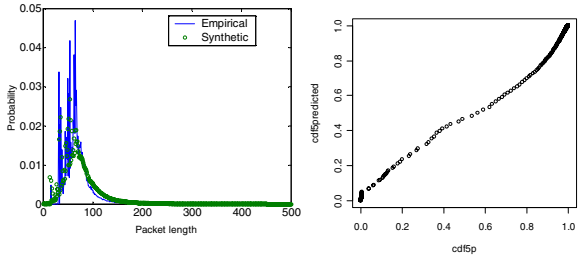

Figure 23. Five player game PMF and Q-Q plot (HL2CS)

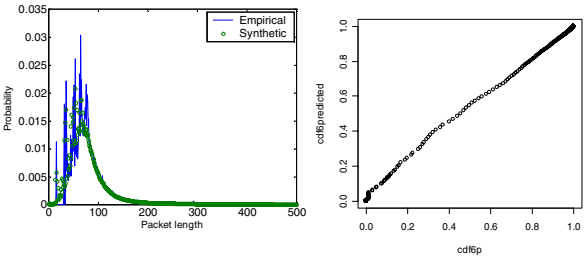

Figure 24. Six player game PMF and Q-Q plot (HL2CS)

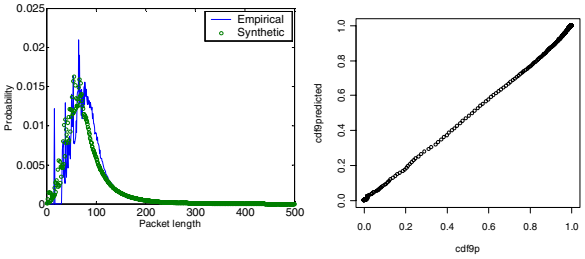

Figure 25. Nine player game PMF and Q-Q plot (HL2CS)

5.2 Validity for games with more than 9-players

We have shown that knowing the PMF of a game with a small number of players, and making some simplifying assumptions, enables reasonably plausible prediction of PMFs for games with larger numbers of players. Now we address the validity of our method for games with more than nine players.

In the absence of empirically derived PMFs we utilize indirect evidence that our approach scales beyond nine players. First we evaluate how the discrepancy between empirical and synthetic PMFs changes as a function of the number of players for N=4 to N=9. Then we extrapolate to discover at what larger value of N the discrepancy might become too large for useful traffic simulations. To do this we need some measure of 'discrepancy'.

For large data sets such as typically occur in teletraffic analysis, traditional statistical tests such as Kolmogorov-Smirnov are inadequate [12]. Instead, to measure discrepancy we use the lambda square measure which is described in detail in Paxson [12] and has been used for FPS game traffic analysis previously [2]. We use the same methodology described by Borella in [2] to determine the discrepancy.

The lambda square test is based on the Chi - squared goodness of fit test but is normalized to allow comparison between distributions where different numbers of samples have been obtained. Of most interest to us is whether or not the lambda square value remains small and largely unchanging as the number of players increases. A sequence of lambda square values that does not significantly increase as the number of players increases suggests that the discrepancy is likely to remain at a similar level as the number of players increases beyond those for which we have empirical data. Consequently,

the technique described is likely to remain valid for larger numbers of players.

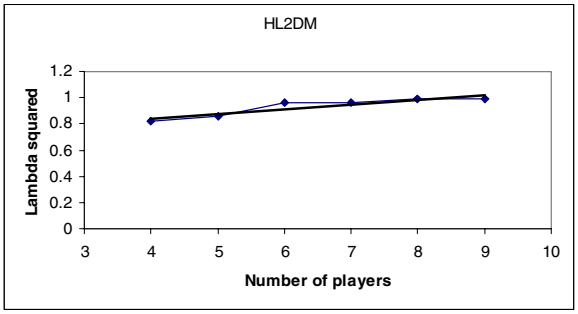

Figure 26. Lambda squared vs number of players for Half-Life 2

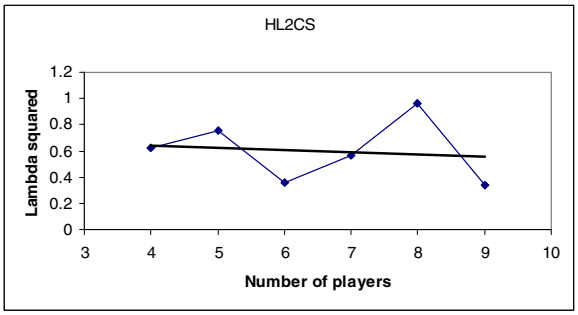

Figure 27. Lambda squared vs number of players for Half-Life 2 Counterstrike

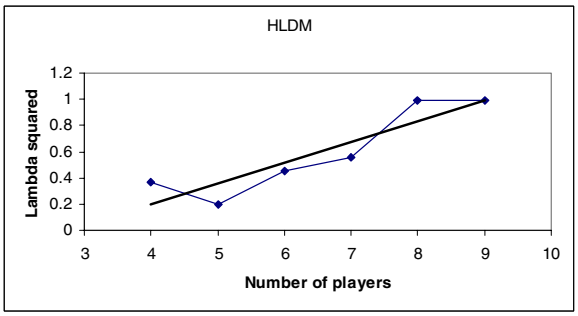

Figure 28. Lambda squared vs number of players for Half-Life

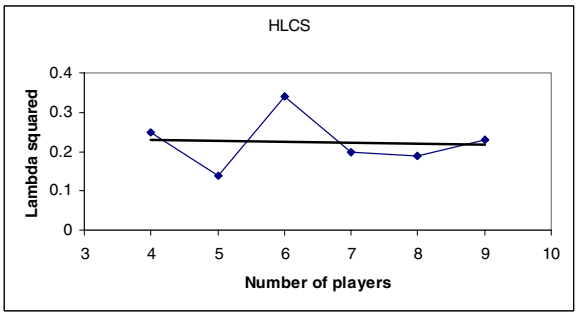

Figure 29. Lambda squared vs number of players for Half-Life Counterstrike

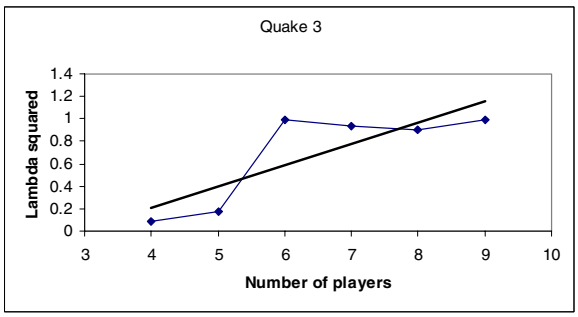

Figure 30. Lambda squared vs number of players for Quake III Arena

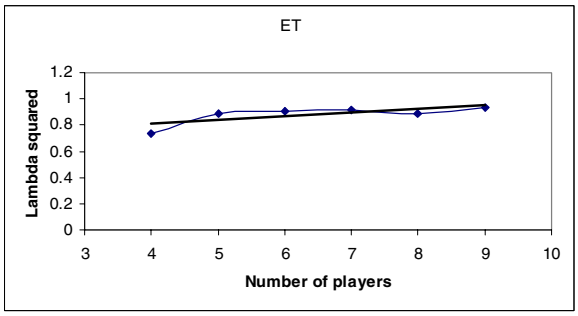

Figure 31. Lambda squared vs number of players for Wolfenstein Enemy Territory

In Figures 26 to 31 we plot the Lambda square value for each game against the number of players. We also include a trend-line showing line of best fit of lambda-squared for each game. From the plots we see that the average discrepancy remains small and, more importantly, the value of lambda-squared does not increase greatly as the number of players increases.

To identify an upper limit on N for synthetic PMF construction we need an upper bound on discrepancy, and hence an upper bound on the lambda-squared value. In practice this upper bound depends on the purpose for which one intends to apply the synthetically derived PMF. When using discrepancy measures to identify suitable models of game traffic, Borella indicated that a value of 1.26 was acceptable [2].

Taking 1.26 as the upper limit, 2- and 3-player empirically derived PMFs would seem likely to produce useful synthetic PMFs up to at least 10 players in Half-Life and Quake III Arena, at least 50 players in Wolfenstein Enemy Territory and Half-Life 2, and up to the game server's physical limits for Half-Life Counterstrike and Half-Life 2 Counterstrike.

The preceding upper bounds on N assume that it is reasonable to perform a linear extrapolation from lines-of-best-fit. The upper bound on N depends on one's tolerance for discrepancy. Both assumptions may be debated when our technique is applied to specific traffic simulation and modeling scenarios. Nevertheless, we believe our technique provides a very useful tool for simulations when empirical data is limited.

5.3 Using convolutions of larger player number games

Our technique is not limited to convolutions of 2- and 3-player PMFs. Where available, it can be beneficial to utilize empirically obtained PMF statistics from games with more than three players. Such PMFs can reduce the number of convolutions required to synthesize PMFs of even larger games.

For example, the server to client PMF of an 18-player game can be synthesized in two ways:

- Five convolutions of an empirically obtained 3-player game PMF
- One convolution of an empirically obtained 9-player game PMF

It is beneficial to minimize the number of convolutions, as the error inevitably introduced by each additional convolution will be reduced.

We illustrate the effectiveness of this approach using our Quake III Arena data to synthesize the server to client PMF of a 9-player game. Figure 32 shows three synthesized PMFs and an empirically derived PMF:

- Three 2-player games and one 3-player game (three convolutions)
- Three 3-player games (two convolutions)
- One 4-player and one 5-player game (one convolution)

We can see that the approximation to the empirical PMF improves as we decrease the number of convolutions involved in the synthesis process. The plot shows three synthesized PMFs along with the empirically obtained data. The synthesized PMFs progressively better approximate the empirical PMF as the number of convolutions used to derive them is decreased.

We can quantify this improvement by calculating the lambda squared value for each synthesis. For the 3- convolution synthesis lambda square is 0.9992, for the 2- convolution synthesis it is 0.9974 and for the 1- convolution synthesis it is 0.9959.

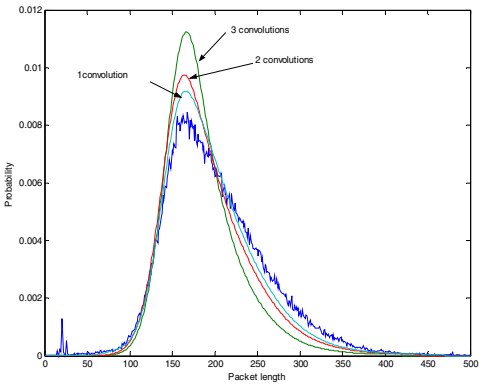

Figure 32. Using fewer convolutions increases the accuracy of the synthesized PMF

6. Future work

Two directions of future research follow naturally from our proposal: testing the generality of the technique for other online games, and establishing the distribution functions for individual packet-to-packet length transitions.

We have shown our approach delivers plausibly useful synthesized PMFs for six different FPS games, without requiring specific knowledge of each FPS game's underlying network code. On this basis we believe that our technique will apply equally well to any FPS game that shares the basic design constraints and goals discussed in section 2 and 3. (For example, Unreal Tournament shows similar empirically derived PMFs [7], and shares similar FPS design goals of minimizing network traffic while keeping all clients appraised of game state changes in a timely manner. Hence we would expect our predictive technique to be applicable.)

Understanding the long-term distribution of server to client packet lengths is important. However, a complete simulation model must also account for the way in which packet lengths change from one packet to the next. Additional study is required into the correct modeling (and synthesis) of packet length transitions on a per-packet basis.

To properly validate the utility of this work the community must ultimately utilize empirical and synthetic PMFs in actual simulations of networks containing a significant mix of game traffic.

7. Conclusion

An important component of simulating multiplayer online game traffic is the packet size distribution of server to client traffic during interactive stages of game-play. Empirically derived statistics are often based controlled trials with small numbers of players. In this paper we focus on first person shooter (FPS) games and introduce a technique for synthesizing packet size distributions based on empirically derived statistics from 2- and 3-player games. This technique extends the utility of empirical datasets derived from controlled trials having a limited number of players (for example, public game traffic trace archives, such as Swinburne University of Technology's SONG database [13]).

Packet statistics are expressed as probability mass functions (PMFs, a discrete equivalent to the probability distribution function). Previously published work has used PMFs to construct analytical expressions for packet distributions. We describe how to synthesize the PMFs of 4-player, 5-player and larger games using convolutions of PMFs derived empirically from 2- and 3-player games.

Using lambda-squared goodness of fit techniques we measured the discrepancy between synthetic and empirical models for up to 9-player games across six different FPS game types. Extrapolating from these discrepancy results leads us to believe that, using only 2- and 3-player PMFs, one can synthesize useful server to client packet size PMFs for games with 10 to 50+ players.

Finally, we have shown how the method can be extended further by using games with more than 3 players as the building blocks

of our syntheses and how this improves the approximation of the synthetic model to the empirical model.

We believe this technique of convolutions is applicable to other FPS games whose network communication protocols share key constraints and design goals.

8. ACKNOWLEDGMENTS

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

9. 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] 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).

[10] 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).

[11] 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).

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

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

[14] 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).