Towards a General Model of First Person Shooter Game Traffic

Philip A. Branch, Grenville J. Armitage
Centre for Advanced Internet Architectures. Technical Report 050928A
Swinburne University of Technology
Melbourne, Australia
{pbranch, garmitage}@swin.edu.au

Abstract- **Although the past few years have seen significant, empirically-based activity in modelling First Person Shooter (FPS) game traffic, there has been only limited work published that explores the underlying reasons for observed FPS traffic characteristics. As a result the generality and scalability of these empirical models is open to question. In this paper we demonstrate how a useful predictive model for FPS traffic patterns can be derived from a number of generic multiplayer game-play requirements. We begin by describing a possible theoretical basis for modelling the message rates and message sizes between server and clients in FPS games. Based on this model we make predictions as to the characteristics of FPS game traffic and then compare the predictions with traffic statistics collected from Quake 3 and Unreal Tournament game trials. Agreement between the predictions from our theoretical model and the statistics from the game trials is very good.**

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

I. INTRODUCTION

Modelling traffic generated by Internet based multiplayer computer games has attracted a great deal of attention in the past few years [4, 5, 7-10, 13-15, 21]. This attention is a consequence of the size of the computer game industry and its growth. It is little appreciated just how large the computer game industry is and how quickly it is growing [1]. Although most industry revenue is from non-networked console games, a significant and rapidly growing part is from online, interactive computer games [16].

In an online interactive game, multiple players at diverse geographical locations interact with each other within a virtual space in real time. In First Person Shooter (FPS) games (one of the most popular genres), the interaction is usually some form of virtual armed combat [12].

As the number of people playing Internet based games increases, it is becoming important that Internet service providers, game server operators and game software developers understand the teletraffic generated by Internet based games. Of particular importance to these groups of people are questions as to how the traffic generated by these games increases as the number of players increases, and how this traffic affects and is affected by, other traffic. To answer questions of this nature simulation and analysis are typically used to model the proposed system before deployment. The simulation or analytical model allows questions involving server and network capacity, and overall system performance, to be answered. However, for this to be effective, good traffic models are needed [11]. It is necessary to understand how the traffic varies as the number of users increases, 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 analysed and modeled, and the models used to predict the consequences for the Internet, it is necessary to analyse game traffic and produce models that can also be used in the same way [6].

This need has resulted in a great deal of work being done in the past few years in modelling game traffic. Traffic models have been developed for popular games such as Quake 3, HalfLife, Counter-Strike and Unreal Tournament [4, 5, 7-10, 13-15]. However, a weakness of this work is that it has been entirely empirical. Traffic model development has involved constructing probability models for the games by examining packet traces and identifying the most appropriate statistical distribution to describe the trace. There have been only the most limited attempts made to understand why the traffic has its particular characteristics. While the empirical models are useful, they are open to criticism regarding their generality and reliability. We may reasonably question whether these models can be generalised to predict the behaviour of other FPS games or scaled to predict the behavior of games with large numbers of players.

In considering the generality of traffic models, it is intriguing to note that the traffic generated by Quake 3, HalfLife, Counterstrike and Unreal Tournament (all popular FPS Internet games) has some interesting similarities. In particular the probability density functions for packet size and interarrival times have a similar shape [5, 10, 14, 15]. The similarity between Quake 3, HalfLife and Counterstrike could be explained by their common lineage, but there is little published evidence to link Unreal Tournament's network engine with that of Quake 3. However, it was possible that respective developers implemented similar ideas given the design constraints of highly interactive games operating over low bandwidth network links.

Scalability of the models is the other important issue. For FPS games, detailed game statistics are usually obtained in controlled trials with only a small number of players (typically less than ten) [13-15, 20]. An important question is whether these models can be extrapolated to much larger numbers of players and if so, how?

In this paper we begin addressing these two issues and in the process, develop a general basis for modelling game traffic. We begin by proposing some simple assumptions of the goals that developers of a FPS Internet game would probably build into their game protocols, develop a simple model of game traffic based on these assumptions, see what traffic profiles the model predicts and then compare the predictions with traffic statistics collected from real Quake 3 and Unreal Tournament game trials.

In attempting to understand traffic generated by FPS games, we need to look at general characteristics of game statistics rather than simple statistics such as mean and variance. So, in this paper, we are more interested in probability distributions of important game traffic statistics than in the actual statistics themselves. In particular we are interested in the rates and sizes of messages transmitted to and from the game server and game clients and whether we can predict, in general terms, common features of these statistics for FPS games, and how these statistics vary as the number of players varies.

We are also interested in broad general characteristics of FPS games. Modern FPS games use quite sophisticated techniques to minimize undesirable network effects [18]. In this paper we make simplifying assumptions which are only approximately correct, but which make analysis tractable.

The rest of the paper is structured as follows. In Section 2 we outline our assumptions and develop a traffic model based on them. In Section 3 we consider packet interarrival times both to and from the server. In Section 4 we model packet lengths from the client to the server. In Section 5 we model packet lengths from the server to the client for two, four, six and eight player games. We see that there is a simple relation between this traffic and the client to server packet length. Section 6 is our conclusion.

In Sections 3, 4 and 5 we compare predictions from our model with actual traffic statistics of Quake 3 and Unreal Tournament from controlled laboratory trials. It is worth noting that agreement between our model and the traffic statistics is very good.

II. MODEL OF FPS GAME TRAFFIC PROTOCOLS

A. *Goals of FPS Protocol Design*

As part of the development of a networked FPS game, the designers need to develop a communications protocol for the exchange of information between players of the game. Its purpose is to specify how information about the state of the game (player locations, current actions, scores, battles and so on) is to be communicated to players. This protocol is distinct from the underlying transport protocol (usually UDP) and should not be confused with it. In this section we attempt to identify the main design objectives of such a protocol.

Most FPS games are server based. An ISP or a game enthusiast runs game server software on an Internet attached machine and invites other players to participate in games running on that server. Players have client software which they also run on an Internet attached device, typically a personal computer. The client software is responsible for reporting the player's actions to the server. The server processes the actions, along with that of other players, and propagates the result (the game-state) to the clients. The client software then uses the result to modify the display seen on the player's computer. So for example, if a player uses a grenade launcher to attack another player, the server decides what the consequences of the attack are and transmits details to the clients. Such details might include the trajectory of the grenade, followed by an explosion and the possible demise of the other player. For the game to operate in real-time and to provide an immersive experience, game-state needs to be distributed at a high rate.

Our first assumption is that in designing a protocol to support such games, a key goal is to minimize traffic across the Internet. Game server operators want as many people as possible to play, including people who have limited bandwidth in accessing the Internet. By minimizing the bandwidth needed for communication between the client and the server, a much larger game player population can be reached. Also, game server operators want to minimize costs. One of their key costs is bandwidth. By keeping traffic to a minimum, cost is reduced. Finally, it is important that the game is fair to all players. If large amounts of data are to be shipped between the client and server, then players on high bandwidth connections will have a significant advantage over players on low bandwidth connections.

Our second assumption is that the protocol will be designed with fairness to all players in mind. To maintain fairness it is important to ensure that all clients have as accurate and timely a copy of the game-state as possible. Since the server contains the definitive copy of the game-state, it needs to distribute it frequently to all players. Also, it needs to distribute it to all players at much the same time interval. To illustrate why this is necessary, consider the case of one player who receives a copy of the game-state every 500 ms and another player who receives it every 50 ms. In 500 ms the second player could engage the first in combat, defeat them, and retreat to safety before the first player even knew they were under attack. The second player has an unfair advantage over the first. Consequently, the rate at which the game-state is received by each client should be frequent and (as much as possible) the same. The same issue of fairness requires that clients report their state to the server at a frequent rate so that it can be propagated to all other players, so that they have an accurate picture of the state of the game.

B. *Consequences of FPS Protocol Assumptions*

The first assumption leads us to expect that game traffic will be largely independent of the client. There may be different versions of client software but the protocol used to communicate between the server and client (and vice-versa) should be independent of the client system. In particular it should deal in actions rather than the detailed delivery of complex, client dependent graphics. In general, the client will transmit short code-words specifying what the client has done. For example a client will transmit a code to the server meaning "Player has jumped left" rather than a detailed video sequence of the player jumping left. The server will then propagate similar short coded messages to other players within that player's field-of-view specifying that this client has jumped left, as well as the consequences of the action. For example, rather than transmitting a video sequence showing the results of a player "jumping left" (such as falling off a cliff, or being shot by another player) the consequences will be coded in as simple a manner as possible and propagated to all clients who have that player in their field-of-view. The client machines will then interpret the code and generate appropriate graphics on the client computer.

If we consider the consequences for client to server teletraffic, we would expect the length of the messages from the client to the server to span a small range of values. The messages would encode the action of the client. Typically, there is a limited number of actions that each client can perform. For example in Quake 3 the number of actions is less than 20 [12]. These include jump left, jump right, turn around, take the object lying on the ground and other similar, simple actions. Consequently, message length from the client to the server will be limited to a small range of values.

A consequence of the second assumption is that we would expect the interarrival time of messages from the server to the client to be, as near as possible, fixed. That is, we would expect to see the server propagate game-state information at roughly a constant rate, leading to an approximately constant message interarrival time.

We would also expect the interarrival time of client to server packets to be similarly fixed for the same reason.

The game-state transmitted from the server to the clients will consist of the state of each player and any possible interactions (battles) between them. We would expect interactions to occur less often than changes in state of individual players. Where there are interactions we would expect the server to need to transmit more information than if there are no interactions.

Since the game-state is propagated to all players, and since each player's actions affect the state of the game, then we would expect the average message length from the server to the client to increase as the number of players increase.

We now use these observations to propose traffic models for each of these packet statistics and compare them with traffic statistics captured from game trials conducted by us in 2003 [15]. The traffic traces are from the previously popular Internet game Quake 3. Quake 3 is a FPS game where players explore a virtual world, collect useful objects and engage in combat with other players [12]. In the trials we set up a server and clients within a laboratory environment. We ran games for set lengths of time using the same game environment with a number of players of different abilities. We collected statistics of packet lengths and interarrival times to and from the game server at the game server machine. Full details can be found in [15].

III. MESSAGE RATES

A. *Client to Server Message Rate*

The server needs to have as accurate information as possible of the location and current action within the game space of each player. This will mean frequent updates from the client to the server of each player's location. The additional requirement of fairness means that updates of client location and activity should be sent by each client at some minimum rate. If one client reports their player's position less frequently than other clients, then there is a potentially unfair advantage for that player. During the time between update messages to the server the player might be able to move large distances, attack other players and then retreat to safety. Fairness dictates that clients should report their location and action at a frequent rate with the possible exception of when the client is inactive.

Consequently we would expect the Probability Density Function (PDF) of the interarrival time of client to server messages to be approximated by an impulse function. In Figure 1 we show the statistics of the interarrival times from the client to the server for Quake 3. Clearly this is not an impulse. Nevertheless, it is a reasonable approximation to one. Empirical models have modeled this distribution with both an impulse and an impulse modified by an exponential distribution [15].

It is worth noting that client to server packet rate is usually configurable by either the client or the server but with a minimum interarrival time. In Quake 3 the default interarrival time is 20 milliseconds but is configurable to a maximum of 50 milliseconds. Generally games will allow the client to report the client state more frequently than the minimum but will not allow the client to report less frequently than some default [12].

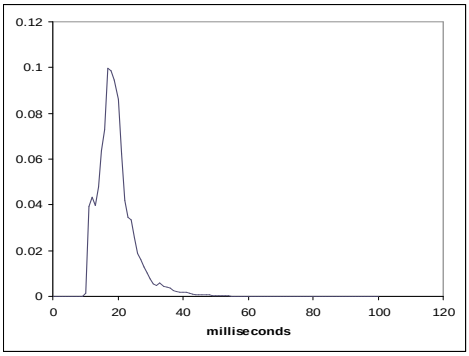

Figure 1. Probability Density Function of Client to server interarrival times measured at the server for Quake 3

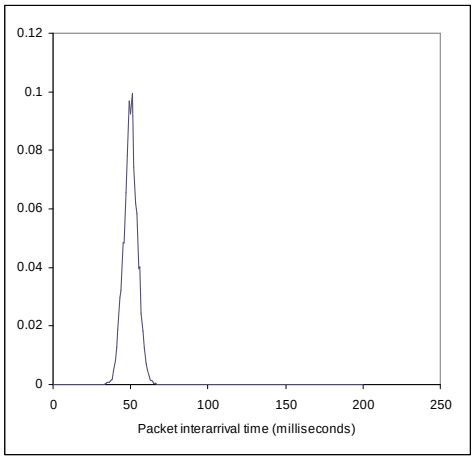

Figure 2. Probability Density Function of Server to client message interarrival times measured at the server for Quake3

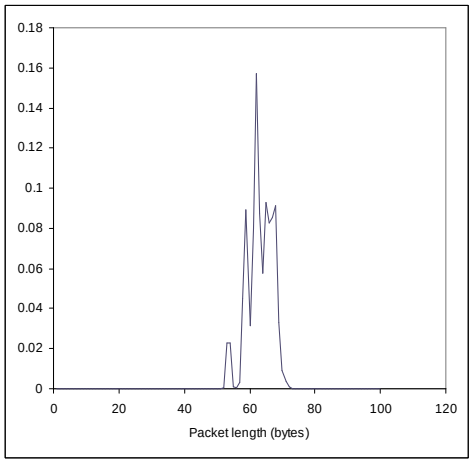

Figure 3. Probability Density Function of Client to server message length measured at the server for Quake 3

B. *Server to Client Message Rate*

Again, applying the fairness assumption we would expect the interarrival time between messages (and hence the message rate) to be close to constant. For the game to be fair to all players the game-state needs to be propagated to all players frequently and (as nearly as possible) at the same rate. Consequently, we would expect the PDF of the server to client message interarrival time to be approximately an impulse function. Figure 2 shows the interarrival time for client to server packets for Quake 3. We see that it is quite a reasonable approximation to an impulse function. Empirical modeling has suggested that this traffic can be modeled with either an impulse or a Laplace distribution [15].

IV. CLIENT TO SERVER MESSAGE LENGTH

We can expect that client to server messages will usually fall into a few different categories, with a few different parameters. The actions that a player can perform, and hence the information that needs to be transmitted back to the server about those actions, will usually be quite limited. It will include information as to where the player is in the game world, how fast they are moving and what actions they are performing. Actions might include “Jump to the left”, “Pick up medical kit”, “Use flame-thrower” and similar. Consequently, we would expect only a few different message types to be sent to the server. Each of these would have a limited range of parameters. As a result we would expect a limited range of message sizes. A further observation is that some of these actions (such as run forward) will occur more frequently than others (such as launch grenade).

Using these observations we propose that the PDF of the client to server message lengths could be modeled by a number of impulse functions spanning a short interval where each impulse function represents a commonly occurring set of actions. We see that the PDF for Quake 3 client to server traffic shown in Figure 3 matches this very well, with a number of impulse functions distributed between 50 and 70 bytes.

V.SERVER TO CLIENT MESSAGE LENGTH

A. *Two Player Game*

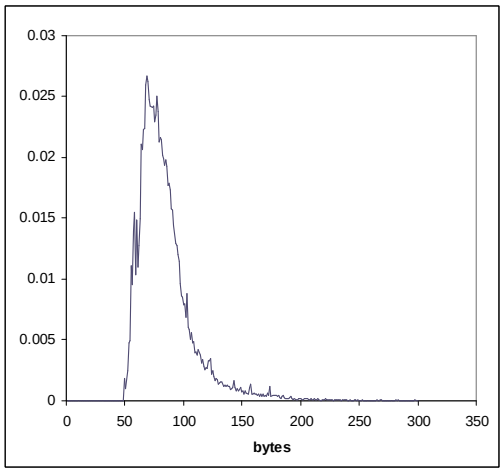

Figure 4. Empirically obtained PDF of the two-player game message length for Quake 3 during active game-play

We now consider server to client packet lengths. We can make a qualitative prediction as to the general shape of the PDF for multi-player games. Propagating details about game-state where players have interacted (shot at each other) will require more data than where players have not interacted. Interactions will not occur all the time. Consequently, we can expect longer messages to occur less frequently than shorter messages, giving rise to a negatively skewed distribution.

In Figure 4 we have plotted the empirically obtained PDF for Quake 3 of the two-player game packet-length from the server to the client. We see that, as predicted, it is a skewed distribution.

We now construct a theoretical model of packet length for two player games. To do so, we need to make some assumptions as to player behaviour and how the game is constructed.

First we assume that each player generates messages whose PDF is similar to that shown in Figure 3.

Second we assume that the traffic generated by each player is independent of other players. That is, we are assuming a homogeneous player population. Although this is obviously open to challenge, we can claim it as a starting point for our analysis and modify it in further refinements.

Third, we assume that the server holds the definitive copy of the game. That is, a player may attempt to carry out an action, but that action (for a variety of reasons) may not succeed. Consequently, we would expect the server to send back to each client information as to what actions the server has recorded the client as having carried out since the last update. We assume this can be modeled by a s PDF similar to that shown in Figure 3.

Fourth we assume that information is sent to each player about the other player's most recent action. Obviously for players to interact they need to know something about the other player's location and behaviour. However, the rise of so-called "wall hacks" has meant that modern FPS games no longer send full details of other player's behaviour to all players. (A "wall hack" occurs where a player cheats by using non-standard client software to interrogate the full game-state to identify the location of players who should otherwise be invisible. An example is where a player is hiding behind a closed door [2]). Consequently, modern FPS games analyse each player's field-of-view and transmit only information that the player is legitimately able to use.

Using these assumptions we can construct a model of server to client packet length for the two player game. We propose that the variable length part of packets from the server to the client are made up of three code-words: a code-word describing the action of the first player, a code-word describing the action of the second player, and a code-word generated by the server describing consequences of player actions, such as explosions, player health points and similar.

The first code-word describes the success or otherwise of the actions the player has transmitted to the server since the server last sent it a copy of the game-state. In modern FPS games there is not necessarily a one-to-one correspondence between the number of packets sent by the client to the server and the server to the client. The default in Quake 3 is for the client to send an update every 20 milliseconds and for the server to send an update every 50 milliseconds. Consequently, the client needs to be informed of whether or not its requested actions have succeeded. We propose modeling this with the PDF of the client to server traffic.

The second code-word describes the actions of the other player. Again, we assume that an approximation to the PDF of length of this code-word is the client to server PDF. However, the effect of limiting the field-of-view of the players will be to negatively skew this PDF. Depending on the game map, each player will, at times, receive little or no information about the other player. Consequently, the code-word describing the other player's action will have an average shorter length than the code-word describing the actions sent to the server by the other client. In other words, the distribution will be skewed. However, as a simplifying assumption we ignore this skewing and assume we can model the information in the second code-word by the same PDF used to model the first code-word. A consequence of this assumption is that our model may under-estimate the number of small packets and over-estimate the number of long packets.

Finally there is the server generated code-word specifying the consequences of player actions. This would include such events as explosions, grenade trajectories and similar. We would expect the length of code-words describing events generated by the server as a result of player actions to have a negatively skewed distribution for the following reasons. The first is as a consequence of efficient coding. If events require the same amount of information to describe then, when an efficient coding scheme is used, the most commonly occurring events are given the shortest code-word, naturally causing a skewed distribution [19]. Also, events requiring large amounts of information to report to the client (such as explosions) occur less frequently than events requiring little information to report (such as changes in player position). Consequently, we propose that the code-word lengths generated by the server as a consequence of player actions can be modeled with a skewed distribution. In the analysis that follows we will model the server generated code-word lengths with an exponential distribution.

Before we begin the analysis we need to note that all packets to and from the server have a fixed header length of approximately 48 bytes. In the analysis that follows we ignore this header.

We formalize our analysis as follows. Denote the random variable describing the length of the variable part of the message length from the server to the client for a two-person game by X. We propose that X is the sum of the lengths of three code-words: C_1, C_2 and C_3. C_1 is the code-word containing the information as to what the server has determined is the client's action since the last update from the server. C_2 contains information as to what the other client has done since the last update. C_3 is the code-word describing the server generated actions.

Denote the random variable describing the variable part of the client to server message length by V. The PDF of V is empirically derived and, for the Quake 3 trials, is shown in Figure 3. We use V to describe the length of both C_1 and C_2. Denote the random variable describing the length of C_3 by Z. Using the assumptions outlined above, it follows that:

$$X = V+V+Z \tag{1}$$

The PDF of the sum of continuous random variables is given by their convolution [3]. If we denote the PDF of X, V and Z by f_X, f_V and f_Z respectively, then the PDF of X is:

$$f_X = f_V * f_V * f_Z \quad (2)$$

where * denotes convolution.

We can also identify some other relationships between X and Z. In particular, since the expectation of the sum of random variables is the sum of expectation of each random variable [3], then from equation 1 the expected value of *X* will be related to *V* and *Z* by

$$\overline{X} = 2\overline{V} + \overline{Z} \quad (3)$$

We have proposed modelling the PDF of Z with an exponential distribution. The exponential distribution is defined by the single parameter λ where 1/λ is the expected value of the random variable [3]. That is:

$$\overline{Z} = \frac{1}{\lambda} = \overline{X} - 2\overline{V} \quad (4)$$

So, in summary, we propose that the variable part of the two player server to client PDF is made up of three code-words: C_1 comprising the server's determination of what the recipient player has done since the last update, C_2 comprising information the recipient player is allowed to know of the actions of the other player since the last update, and C_3 generated by the server and comprising consequences of actions and interactions between the two players. We assume that the length of C_1 can be modeled by *V*. We assume that the length of C_2 can also be modeled by *V*. But in making this assumption we have assumed that field-of-view restrictions have a negligible effect on the length of C_2. Finally we assume that the length of C_3 can be approximated by a negative exponential of the form:

$$f_Z = \lambda e^{-\lambda t} \quad (5)$$

In the next two sections we compare the predictions derived from this analysis with Quake 3 and Unreal Tournament statistics.

In the next two sections we compare the predictions derived from this analysis with Quake 3 and Unreal Tournament statistics.

B. *Modeling the Quake 3 Two Player Game*

We now use the analysis in the previous section to predict the PDF of the Quake 3 two player game.

We need to specify the value of λ to use in this analysis. If we know the mean value of the variable part of the packet length for the two player game (that is, $\overline{X}$) then we can use it to determine $\overline{Z}$ through equation 4. In the Quake 3 trials $\overline{X}$ is approximately 47 bytes and $\overline{V}$ is approximately 11 bytes. Consequently, using equation 4, $\overline{Z}$ is approximately 25 bytes. So the value of $_1$ that we will use is 1/25.

At first glance it might appear that we are assuming the result ($\overline{X}$) that we are setting out to find. However, we are not doing so. Our theoretical model proposes that an exponential distribution might be suitable for modelling *Z*. We have made use of equation 1 to attempt to determine what exponential distribution might be suitable. Also, we have made no assumptions regarding the PDF of *X*, the determination of which is our main goal.

So we propose that:

$$f_Z = \frac{e^{-t/25}}{25} \quad (6)$$

Using equation 2 we can now determine values for f_X. The results are shown in Figure 5 along with the empirically derived PDF (denoted by EPDF).

The agreement between the PDF derived from our theoretical modelling and the captured statistics is very good. We have made a number of simplifying assumptions in this analysis about player behaviour and the consequences of field-of-view restrictions. Even so we see good agreement between our model and the empirically derived model for Quake 3. This gives us some confidence that our assumptions are reasonable. However, it is interesting to note that the theoretical model predicts more longer packets and fewer shorter packets than were observed in the empirical model. As noted earlier this is consistent with our approximating the PDF of the length of C_2 with *V*. Taking into account the field-of-view constraints that modern FPS games impose, it might be more accurate to model the length of C_2 with a skewed version of *V*. However, the difference is also consistent with the Exponential distribution overestimating the length of C_3. Modeling the length of the code-words C_1, C_2 and C_3 is an area for future research.

In the empirically obtained data, there is a large spike at 48 bytes. This appears to be caused by a keep-alive message transmitted when players become idle. Since we are interested in the traffic generated by active periods of play, these are not included in the plots.

We can quantify how good a match fx and EPDF are by using the Kolmogorov-Smirnov test (K-S). The K-S test is used to test whether two sample data sets come from the same distribution, without the distribution needing to be specified [17]. The range of packet sizes used in the analysis is 1 to 300.

Table 1. K-S test for two player games

Maximum value of $\lvert F_0(X) - S_n(X)\rvert$	Maximum value of $\lvert F_0(X) - S_n(X)\rvert$ for level of significance	
	0.20	0.15
0.0545	0.0617	0.0658

From the test we see that we can accept the hypothesis that the predicted and observed statistics are from the same distribution to a very high level of significance.

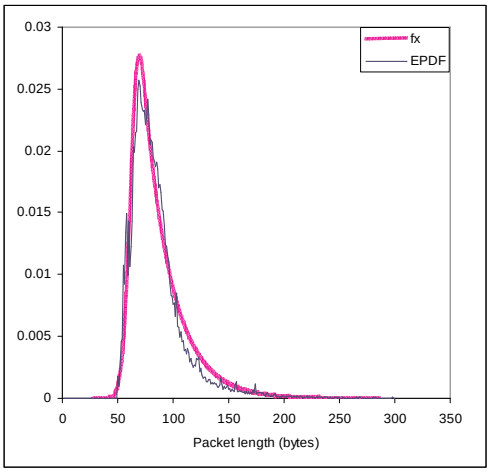

Figure 5. Predicted (fx) and observed (EPDF) two-player game message length PDFs for Quake 3 during active game-play

C.*Modeling the Unreal Tournament Two Player Game*

We now compare the Unreal Tournament statistics from our game trials with predictions from the above analysis. First we need to know V. From the Unreal Tournament game trials, the PDF of V is shown in Figure 6.

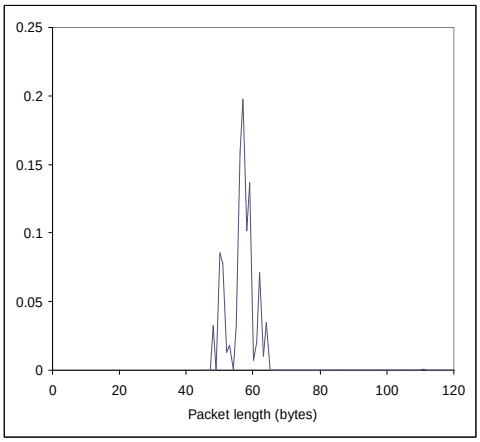

Figure 6. Probability Density Function of Client to server message length measured at the server for Unreal Tournament

The next step is to obtain the average packet length for the two player game. In this case it is 42. Consequently, we model the length of the server generated code word by

$$f_Z = \frac{e^{-t/42}}{42} \qquad (7)$$

Using the same analysis as before the empirically derived (EPDF) and predicted (fx) PDFs are shown in Figure 7.

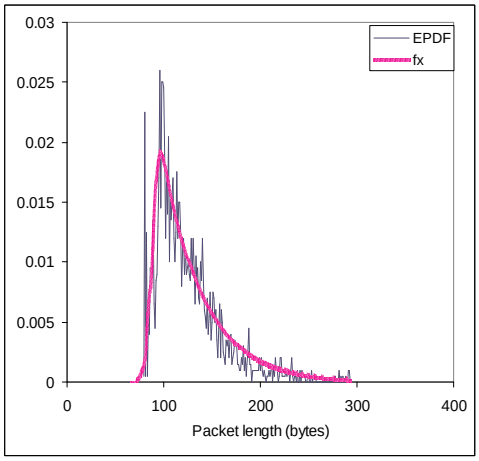

Figure 7. Predicted (fx) and observed (EPDF) two-player game message length PDFs for Unreal Tournament during active game-play

Once again we see good agreement between the empirically derived and predicted models. However, there is clearly much more complexity in the empirical data for Unreal Tournament than there is for Quake 3. We can determine how good a match the two distributions are by using the K-S test. Details of the test are shown in Table 2. For this data set the sample size is 500.

From the K-S test we can see that we have excellent agreement between the two distributions.

Table 2. K-S test for Unreal Tournament two player games

Maximum value of $\lvert F_0(X) - S_n(X) \rvert$	Maximum value of $F_0(X) - S_n(X)$ for level of significance	
	0.20	0.15
0.0333	0.0478	0.05098

D.*Games With More Than Two Players*

We now present an analysis of packet length for games with more than two players. To carry out this analysis we use the same assumptions we made in analyzing the two-player game. In particular, we assume that player behaviour is homogeneous in that player actions can be described by the same random variable *V*, regardless of the number of players.

Since game-state is made up of the aggregate behaviour of players and code-words generated by the server as a consequence of player behaviour, we would expect the average message length to increase as the number of players increases. If we assume players are homogenous in their game-play then as the number of players increases the information that makes up the game-state can also be expected to increase linearly. Figure 8 shows the mean packet length for two, four, six and eight player games derived from statistics collected

from our Quake 3 trials. As predicted we see that the mean of the packet length increases linearly.

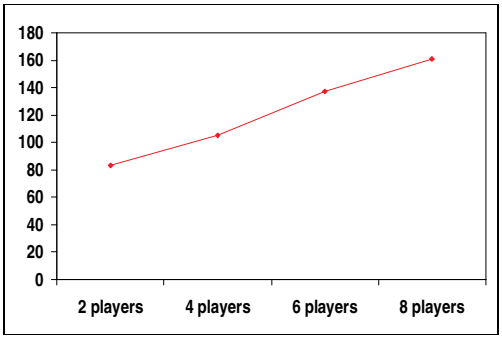

Figure 8. Mean packet length Quake 3

We can also predict the PDF of packet length for larger numbers of players. We know the PDF of a two player game. We now use it to predict the PDF of games with more than two players. The PDF of a two player game gives information about the behavior of each player in isolation and of their interactions. If we assume that most interactions (battles) remain between two players and occur at approximately the same rate for each player regardless of the individual player and the number of players (our homogeneity assumption), then we can construct a probability model of games with more than two players in the same way that we constructed a probability model for the two player game. Again we denote the variable part of the message length of a two player game by the random variable X and its PDF by fX . As before we assume player behavior homogeneity. In particular, we assume that each player engages in the same number of battles regardless of the number of players, and so contributes at the same rate to server generated code-words. Using these assumptions, the message length of a four player game will be $X + X$, that of a six player game $X + X + X$ and that of an eight player game $X + X + X + X$ and so on. Also, since the PDF of the sum of two or more random variables is the convolution of their PDFs, the PDF of four, six and eight player games is respectively:

$$f_{\text{four player game}} = f_{X+X} = f_X * f_X \quad (8)$$

$$f_{\text{six player game}} = f_{X+X+X} = f_X * f_X * f_X \quad (9)$$

$$f_{\text{eightplayergame}} = f_{X+X+X+X} = f_X * f_X * f_X * f_X \quad (10)$$

where * denotes convolution.

Figure 9, Figure 10 and Figure 11 show the predicted (fx) and empirically derived PDFs (EPDF) of four, six and eight player games respectively from the Quake 3 trials, while Figure 12 shows the same results for the four player Unreal Tournament game. Once again we see excellent agreement between the predictions of our model and the statistics collected from the Quake 3 and Unreal Tournament trials.

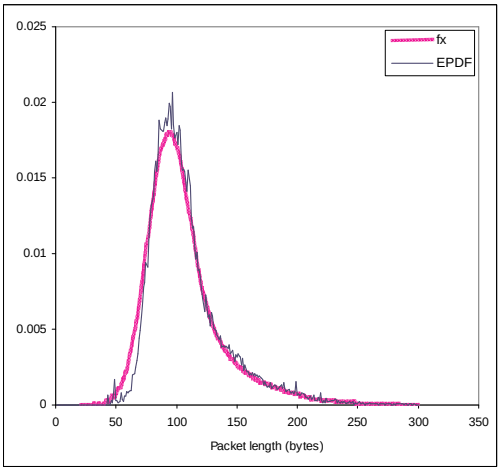

Figure 9. Predicted (fx) and observed (EPDF) four-player game message length PDFs for Quake 3 during active game-play

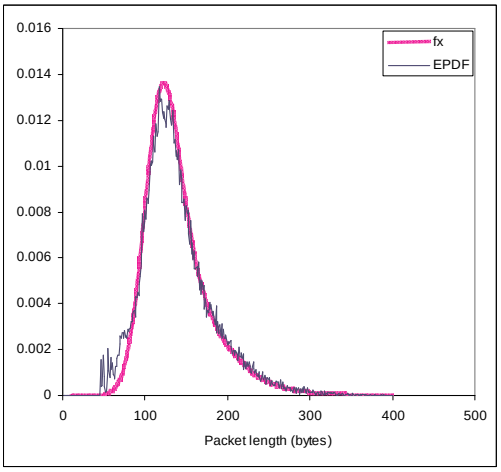

Figure 10. Predicted (fx) and observed (EPDF) six-player game message length PDFs for Quake 3 during active game-play

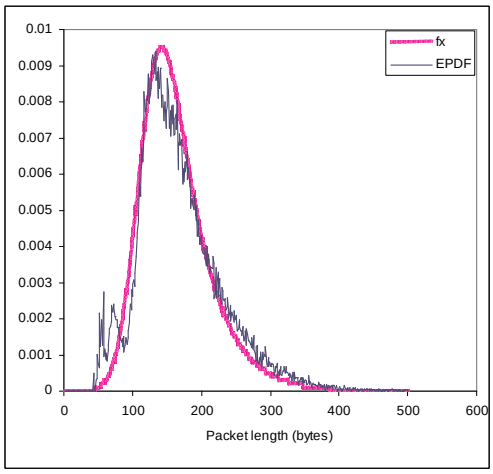

Figure 11. Predicted (fx) and observed (EPDF) eight-player game message length PDFs for Quake 3 during active game-play

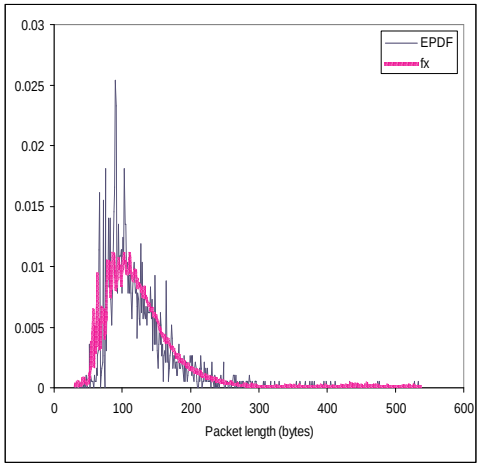

Figure 12. Predicted (fx) and observed (EPDF) four-player game message length PDFs for Unreal Tournament during active game-play

Again we can use the K-S test to quantify the agreement between the predicted and empirically derived distributions [17]. The range of the variable parts of the packet sizes for the four-player, six-player and eight player Quake 3 games are, respectively 250, 350 and 450 bytes. The details of the test for levels of significance at 0.2 and 0.15 are shown in Table 3. From the table, we can see that for the four and six player games, we can accept the hypothesis that the predicted and observed statistics are from the same distribution to a very high level of significance. For the eight player game, the level of significance is lower, but we can still accept that the two data sets come from the same distribution. Table 4 shows the same analysis for the four player Unreal Tournament game. The K-S test shows that we can accept that the two data sets come from the same distribution to a high level of significance.

Table 3. K-S test for Quake 3 four, six and eight player games

Number of Players	Maximum value of $\|F_0(X) - S_n(X)\|$	Maximum value of $F_0(X) - S_n(X)$ for level of significance	
		0.20	0.15
4	0.0435	0.0676	0.0721
6	0.0525	0.0572	0.0609
8	0.0513	0.0478	0.0537

Table 4. K-S test for Unreal Tournament four player game

Number of Players	Maximum value of $\|F_0(X) - S_n(X)\|$	Maximum value of $F_0(X) - S_n(X)$ for level of significance	
		0.20	0.15
4	0.0435	0.0474	0.0505

Our model of server to client message sizes enabled us to make a number of predictions about packet length distributions. It predicted that the distribution would be negatively skewed. It predicted that the mean length of the messages would increase linearly as the number of players increased. Using simplifying assumptions as to player homogeneity and field-of-view restrictions, it showed how the PDF of message length for a two player game could be derived from the client to server PDF. Finally it showed how the PDF of message length for games with more players could be derived from the PDF of a two player game. All these predictions are well supported by the data from both Quake 3 and Unreal Tournament trials.

However, despite the generally excellent agreement there is some evidence that the assumptions of player homogeneity and field-of-view limitations might be less valid for larger numbers of players. Agreement between the predicted and observed packet lengths for the eight player game, while still very good, is not as good as for the games with smaller numbers of players. The empirical data has fewer shorter packets and more longer packets than our model predicts. This would be consistent with one or both of the homogeneity or field-of-view assumptions breaking down. As the number of players increases, proportionally more battles are likely to occur. Also as the number of players increases, restrictions on field-of-view are likely to occur more often. Understanding how the assumptions of player homogeneity and field-of-view break down and their effect on game traffic are additional areas for future research.

We can make a number of observations about game traffic based on this analysis. The first is that the same analysis can be successfully applied to two FPS games with different development histories.

The second is that aggregate traffic generated by the server will increase according to the square of the number of players. In the absence of multicast there is a single virtual connection to each player. As shown in Figure 8 the length of the packet containing the game state increases linearly as the number of players increases. So the total traffic delivered by the server increases according to the square of the number of players.

The third is that since the packet length for larger numbers of players can be described as a multiple of the random variable describing the two player game we would expect (from the Central Limit Theorem) the packet length distribution to trend towards a Normal distribution [3].

Finally, we see no evidence of anything likely to produce fractal behaviour in traffic rates, which is in agreement with the observations made in [10].

VI. CONCLUSION

The main contribution of this paper is that it puts the large body of empirically derived game traffic models onto a firm theoretical foundation. Our theoretical model of the Interarrival time and packet size of traffic to and from the server has predicted results that agree well with

the empirical data from our Quake 3 and Unreal Tournament trials. However, it is important to note that this paper is not an attempt to explain traffic generated by those particular games. Rather we have attempted to predict what traffic a generic FPS game is likely to produce and used traffic traces of Quake 3 and Unreal Tournament to test our predictions. Our concern is with understanding the general characteristics of FPS game traffic, rather than understanding any particular game. Our ultimate goal is to be able to produce game traffic models that we can trust for use in network simulations. We need to know what can and cannot be generalized about the empirical models. The work in this paper has given use some insights into how game traffic is constructed and so given us some confidence in judging what we can and cannot generalize about the empirical models.

At the start of this paper we posed two questions about the empirical models of game traffic that have been developed over the past few years: “Can they be generalized?” and “Are they scalable?”

We began to answer these questions by proposing a number of criteria that Internet-based FPS games would probably need to meet. We then made a number of predictions as to the characteristics of the traffic such a protocol would generate and formulated these as a theoretical traffic model. Finally, we compared our theoretical model with empirical data captured from controlled laboratory-based game trials. We have seen excellent agreement between the predictions of the theoretical model and the empirically derived model.

Our analysis began by modeling message interarrival time. We then analysed message length PDFs. We showed how server to client message lengths for a two player game could be modeled, and demonstrated how to predict message lengths for games with more than two players. Despite having to make some simplifying assumptions, our analysis agreed very well with the empirical data. It showed that there are some quite simple relationships between client to server traffic, server to client traffic for two player games, and server to client traffic for games with larger numbers of players.

This approach to modelling game traffic opens up a number of research areas. We have made a number of assumptions regarding protocol design and player behaviour. Since predictions based upon them agree very well with empirically obtained data this would seem to provide strong evidence that they are reasonable. However, further work in verifying them or showing in what situations they are invalid needs to be done. In particular the assumptions of player homogeneity need to be investigated. How valid is the assumption of player homogeneity and what are the effects of it breaking down?

In our analysis of the two player game we have made simplifying assumptions about how field-of-view restrictions affect the game-state distributed to players. While the simplifying assumptions seem to result in a good prediction of the Quake 3 and Unreal Tournament traffic profiles there is much work to be done in identifying if, how, and in what way the assumptions fail. For example, how does a map's virtual geography interact with field-of-view restrictions influence the information transmitted to each client?

We have analysed packet interarrival times and packet lengths as though the distributions are constant throughout periods of game-play. Future work should investigate how these network characteristics vary with time.

Single-server first person shooters are not the only games becoming prevalent on the Internet. Future work should consider whether peer-to-peer and multi-server games can be analyzed as we have analyzed single-server games, and whether similar assumptions can be formulated for other game styles.

In this paper we have made a step towards more accurately understanding and predicting online first person shooter game traffic. We have shown that it has a predictable structure and, as a result, we can judge the generality and scalability of some of the empirical models that have been developed in the past few years. Consequently, we can have more confidence in simulation and analytical models used to investigate and design game traffic systems than would otherwise be the case.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Console wars The Economist, 2002.

[2] Wikipedia http://en.wikipedia.org/wiki/Wallhack, 23 August 2005

[3] Ash, R. Basic probability theory. Wiley, New York, 1970.

[4] Borella, M., Source models of network game traffic. in Proc. Networld+interop'99, (Las Vegas, NV, May 1999).

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

[6] Cunha, C., Bestavros, A. and Crovella, M., Characteristics of WWW Client-based Traces. Boston University Technical Report, BUCS-95-010 (1995).

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

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

[9] 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).

[10] 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).

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

[12] idsoftware. Quake 3 http://www.idsoftware.com/, February2005

[13] 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).

[14] 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).

[15] Lang, T., Branch, P. and Armitage, G., A synthetic model for Quake 3 traffic. in Proc. ACM SIGCHI Advances in Computer Entertainment (ACE2004), (Singapore, June 2004).

[16] McCreary, S. and claffy, k., Trends in wide area IP traffic patterns: a view from Ames Internet Exchange. in 13th ITC Specialist Seminar on Measurement and Modeling of IP Traffic, (Monterey, California, September 2000).

[17] Sprent, P. Applied nonparametric statistical methods. Chapman and Hall, London, 1989.

[18] The Valve Developer Community. Source Multiplayer Networking http://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking, 15 August2005

[19] Wells, R. Applied Coding and Information Theory for Engineers. Prentice-Hall, New Jersey, 1999.

[20] Zander, S. and Armitage, G., Empirically measuring the QoS sensitivity of interactive online game players. in Proc. Australian Telecommunications, Networking and Applications Conference 2004 (ATNAC 2004), (Sydney, December 2004).

[21] 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).