An ARMA(1,1) Prediction Model of First Person Shooter Game Traffic

Philip A. Branch, Antonio L. Cricenti, Grenville J. Armitage

Centre for Advanced Internet Architectures
Swinburne University of Technology
Melbourne, Australia
pbranch@swin.edu.au
tcricenti@swin.edu.au
garmitage@swin.edu.au

***Abstract*—Modeling 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 need to simulate correctly the network impact of highly interactive online game genres such as the first person shooter (FPS). Packet size distributions and autocovariance models are important elements in the creation of realistic traffic generators for network simulators. In this paper we present simple techniques for creating representative models for N-player FPS games based on empirically measured traffic of 2- and 3-player games. The models capture the packet size distribution as well as the time series behaviour of game traffic. We illustrate the likely generality of our approach using data from seven FPS games that have been popular over the past nine years: Half-Life, Half-Life Counterstrike, Half-Life 2, Half-Life 2 Counterstrike, Quake III Arena, Quake 4 and Wolfenstein Enemy Territory.**

I. INTRODUCTION

Modeling traffic generated by Internet-based multiplayer computer games has attracted a great deal of attention in the past few years [1-14]. 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 User Datagram Protocol (UDP) over IP for transport and do not adjust packet rates in response to 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 [15]. This has motivated research community interest in predicting and simulating the traffic load imposed on network links by multiplayer FPS games.

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 [16]. This paper improves on the time series behaviours and packet size distributions predicted by existing game traffic models.

Understanding how game traffic varies as the number of players 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. Web and other traffic has been analyzed and modelled and the models used to predict the consequences for the Internet [17]. It is now desirable to analyze game traffic and produce models that can be used in the same way.

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. However, traffic in the server to client direction usually shows distinct variation as the number of players increases [6]. 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 have been created that match the statistical packet size distributions for each N-player game, for N = 2, 3, and so on. Yet some public FPS game servers may be configured to allow 50+ players [15]. Controlled collection of empirical data for games with such large numbers of players is challenging. There is a need for techniques that allow extrapolation of statistical characteristics from games with small numbers of players to games with much larger numbers of players.

Since the initial work by Borella [1], FPS game traffic has usually been modelled by examining empirical packet traces and fitting an appropriate standard distribution to the observations. However, a major shortcoming of this approach is that the correlation between successive packet lengths is not retained as the packet payload lengths are simply drawn from the appropriate distribution.

There has been some limited work that attempts to model the correlation between packet lengths in FPS games. Branch et al. [18] used a Discrete Markov Chain to model the server to client acket size distribution of a two player game. The resulting Markov model can then be used to predict the statistics of an N-player game. The resulting models do predict the distribution of packet payload size for differing numbers of players and capture some of the autocorrelated behaviour. However, the models only produce packets whose sizes are integer multiples of the median packet length of a two player game. In practice the packet length distribution is

978-1-4244-2295-1/08/$25.00 © 2008 IEEE

not limited in this way. Also, the autocorrelated behaviour is captured only approximately. Further work by Cricenti et al showed that the ARMA(1,1) model captured the time-series behaviour of FPS game traffic well [19]. We use the ARMA(1,1) model in this work.

In this paper we propose and illustrate a technique for extrapolating N-player traffic statistics from empirically measured traffic of 2- and 3-player FPS games. The extrapolated traffic sequences (based on ARMA(1,1) models) both capture the time series behaviour of the game traffic as well as the packet length distribution. This allows small-scale empirical measurements (for example, from public game traffic trace archives such as Swinburne University of Technology's SONG database [14]) to be applied to larger scale simulations of FPS traffic acting on an IP network. We illustrate the potential generality of our approach using data from seven FPS games released between 1998 and 2006: Half-Life, Half-Life Counterstrike, Quake III Arena, Quake 4, Wolfenstein Enemy Territory, Half-Life 2 and Half-Life 2 Counterstrike.

The paper is structured as follows. Section II discusses the nature of FPS game traffic and explains why server to client packet lengths are of most interest. Section III introduces Box-Jenkins time series analysis and the ARMA(1,1) model. Section IV shows how prediction models can be constructed based on ARMA(1,1) models for large numbers of players when data for games with only small numbers of players is available. Section V presents results that show there is good agreement between packet lengths generated synthetically using this technique and packet lengths measured empirically from game trials. Section VI is our conclusion

II. First Person Shooter Games

Multiplayer online games have an underlying requirement that game-state information is shared amongst all players in near 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. Most FPS games use a client-server model (including the seven 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 and their consequences. 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.

A. 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. (Games can also be run on a private, local IP network – commonly referred to as 'LAN parties'.) The game client updates and renders the game's virtual world on the client's screen based on messages received regularly from the game server. User inputs to the game client (actions such as walking, exploring or shooting weapons) are passed to the game server to be verified and the consequent changes to game state (health points, explosions, etc) propagated to other players.

Game-state updates must occur in a timely manner, with minimal bias towards any particular player. In FPS games, timeliness is achieved by sending a unicast IP packet to each client at fixed intervals, 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 [11], [12]. Each client receives an update packet every interval regardless of how much in-game activity is occurring.

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 40ms [11], [12].

B. First Person Shooter Game Traffic

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.

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 [1, 6].

On the other hand, 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 [1, 6].

The in-game activity conveyed in a single update packet includes a component containing information that is proportional to the number of other players visible to a client at that 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 the virtual game world [15].

It is sometimes suggested that instead of using Stochastic methods, why not examine the source code, in the hope that it will provide insight into the nature of game traffic generated by the server? There are two responses to this. First, source code is usually not available for newly released games (limiting our ability to model game traffic via code analysis while the games are still popular). Second, even when available, source code provides very little information as to the game traffic generated. Examination of the Quake III Arena source code shows that the traffic generated is highly compressed and that the server sends only information about objects in the receiving player's field of view. However, in attempting to determine the nature of the traffic generated by the server, all that a close examination of the code reveals is that the server output is driven by its input; that is the behaviour of the players. Consequently, since an individual player's behaviour has a great deal of unpredictability in it, understanding the server code tells us little about the traffic generated by the server and Stochastic methods are more appropriate for understanding server traffic.

C. *Phases of Game-play and Game Traffic*

In most FPS games there are three phases of interaction between client and server that impact on network traffic.

- A client 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 the game is in progress (players run around the virtual world, shooting or otherwise 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 a player wins the previous 'round').

Tight control over network jitter and packet loss is only essential 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 affecting the player's game experience.

III. ARMA(1,1) Model of FPS Game Traffic

In [19] Cricenti et al showed that FPS server to client traffic is well modelled by ARMA(1,1) models. We briefly review ARMA models before demonstrating how models obtained for 2- and 3-player games can be used to predict the traffic behaviour of games with larger numbers of players.

The zero mean ARMA(1,1) model is:

$$X_t = \phi_1 X_{t-1} + \theta_1 Z_{t-1} + Z_t \qquad (1)$$

The residuals Z_t in (1) are independent identically-distributed random variables with zero mean.

The autocovariance function (ACF), of a random process describes the similarity of the process with itself at different points in time (lag). Thus the ACF can be considered as a measure of the "memory" of the process. Generally speaking if the ACF dies off quickly then the process is stationary.

Autocovariance functions are just one of the useful tools in identifying appropriate ARMA models. Other tools are partial autocorrelation functions and measures such as the Akaike Information Criterion [20]. Fig. 1 shows the normalised AIC and a partial ACF for Quake III Arena. The AIC suggests that an AR(1) model might be an acceptable time-series model for this traffic. An AR(1) process is a special case of the ARMA(1,1) process with $\theta = 0$. However the Partial Autocorrelations do not go to zero sufficiently quickly to warrant a simple AR(1) model. Consequently it may be that an ARMA model is more appropriate. Given that the AIC shows there is little to choose between an ARMA(1,1) and higher order models, we chose to model the traffic with an ARMA(1,1) model. It is worth noting that this results holds for all the games we analysed. For a fuller treatment of this topic see [19].

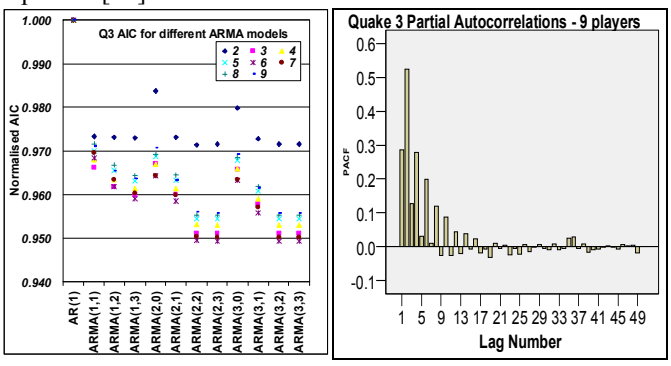

Fig. 1 AIC Criterion abd Partial ACF for Quake III Arena

IV. Prediction Models based on ARMA(1,1)

We now show how ARMA(1,1) models for 2- and 3-player games can be used to predict ARMA(1,1) models for N-player games, where N ≥.4.

We make a number of assumptions in our analysis:

- The nature of game play for individual players does not change significantly 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 behaviour. 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 time series behaviour of an N-player game can be constructed through adding together the random variables describing the time series behaviour 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 behaviour, we propose that, for example, the random variable describing the time series behaviour of 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 time series behaviour of games with larger number of players. We now formalize this analysis.

A 2-player game (zero mean) is described by the ARMA(1,1) sequence:

$$X_{2,t} = \phi X_{2,t-1} + \theta Z_{2,t-1} + Z_{2,t} \quad (2)$$

Consider a 2-player game sequence generated by the above model

$$X^{(1)}{}_{2,t} = \phi X^{(1)}{}_{2,t-1} + \theta Z^{(1)}{}_{2,t-1} + Z^{(1)}{}_{2,t} \quad (3)$$

and another 2-player game sequence

$$X^{(2)}{}_{2,t} = \phi X^{(2)}{}_{2,t-1} + \theta Z^{(2)}{}_{2,t-1} + Z^{(2)}{}_{2,t} \quad (4)$$

One of our assumptions is that player behaviour is similar. Consequently, we would expect two different games to be described by the same parameters φ and θ.

Based on our assumptions we can predict that a typical four player game can be described by

$$\begin{aligned} X_{4,t} &= X^{(1)}{}_{2,t} + X^{(2)}{}_{2,t} \\ &= \varphi(X^{(1)}{}_{2,t-1} + X^{(2)}{}_{2,t-1}) + \theta(Z^{(1)}{}_{2,t-1} + Z^{(1)}{}_{2,t-1}) + Z^{(1)}{}_{2,t} + Z^{(2)}{}_{2,t} \end{aligned} \quad (5)$$

We note that this function has the form:

$$X_{4,t} = \varphi X_{4,t-1} + \theta Z_{4,t-1} + Z_{4,t} \quad (6)$$

That is, it is an ARMA(1,1) model with the same values of φ and θ as the 2-player game. In other words, we predict that φ and θ should be constant for a particular game with an even number of players, regardless of the number of players. We can do a similar analysis for games with an odd numbers of players. In Section V we show that these parameters are largely unchanged as the number of players increases.

If we denote the probability mass function of Z_2 by f_{Z2} and Z_4 by f_{Z4} then our modelling predicts that f_{Z4} can be obtained from a convolution of f_{Z2} and itself. That is:

$$f_{Z4} = f_{Z2} * f_{Z2} \quad (7)$$

In a similar way we can obtain ARMA(1,1) models based on 2- and 3- player games for N-player games where N is an odd number. For the 5-player game, we can predict that it can be described by:

$$X_{5,t} = X^{(1)}{}_{2,t} + X^{(1)}{}_{3,t} \quad (8)$$

In the next section we evaluate the effectiveness of this approach in predicting the PMF and autocovariance functions for games with $N \geq 4$.

V. Results

The key result we present in this section is the application of the previous analysis to predict the PMF and ACF of server to client packet length for the seven FPS games we are analysing. For reasons of space we only show plots for a selection of games, but similar analysis has been carried out for all seven of the games with number of players ranging from 4 to 9. The empirical results were obtained from game sessions of approximately 20 minutes duration each, generating approximately 20,000 samples for each player.

A. Plots of Probability Mass Functions and Autocovariance Functions

Plotted on the graphs below are the predicted PMFs and ACFs, and their corresponding empirical values obtained during game play trials. In all cases the agreement between the predicted and empirical results is satisfactory.

We obtained the plots with the following steps:

- Capture statistics of traffic during active game play.
- Determine φ and θ for the 2- and 3-player games
- Determine the residuals for the 2- and 3-player games
- Determine the PMF of the 2- and 3-player residuals
- Take the necessary convolutions of the 2- and 3-player game residuals PMFs to construct synthetic residual PMFs of games with larger numbers of players
- Generate a sequence of packet lengths using the ARMA(1,1) constants and the synthetic residuals using Equations (5) and (8)
- Extract the PMF and ACF from the synthetic ARMA(1,1) sequence and extract the PMF and ACF from the empirical sequence and plot on the same set of axes.

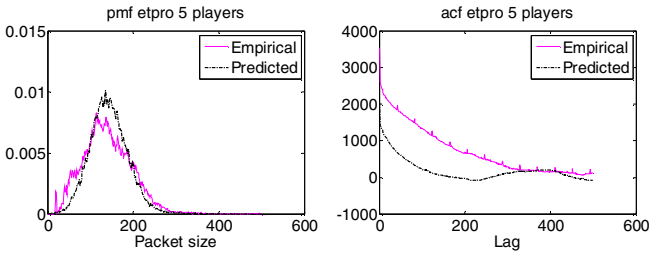

Fig. 2. Wolfenstein Enemy Territory Probability Mass Function and Autocovariance Function for 5 Player Game

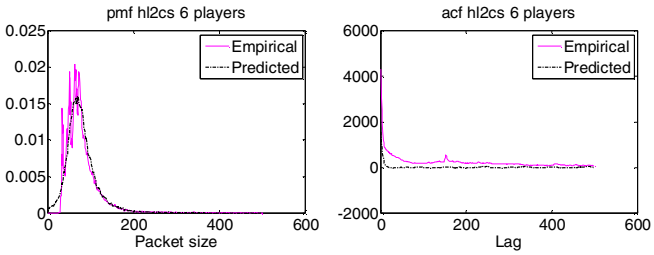

Fig. 3. Half-Life Counter Strike 2 Probability Mass Function and Autocovariance Function for 6 Player Game

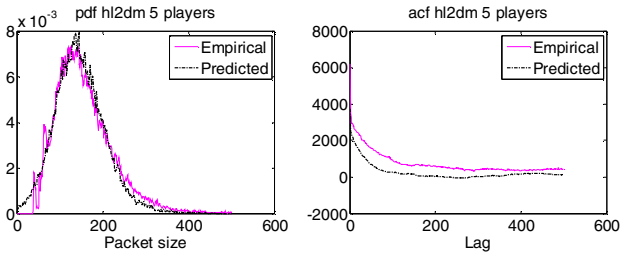

Fig. 4. Half-Life Death Match 2 Probability Mass Function and Autocovariance Function for 5 Player Game

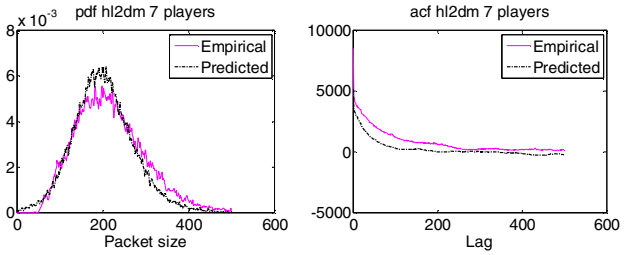

Fig. 5. Half-Life Death Match 2 Probability Mass Function and Autocovariance Function for 7 Player Game

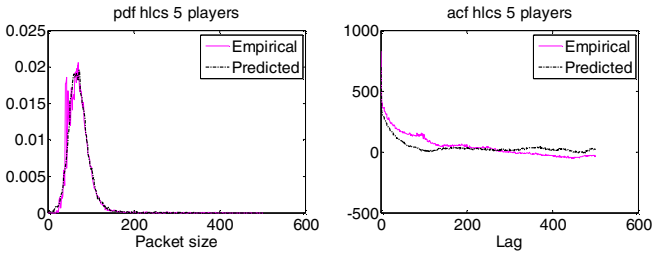

Fig. 6. Half-Life Counterstrike Probability Mass Function and Autocovariance Function for 5 Player Game

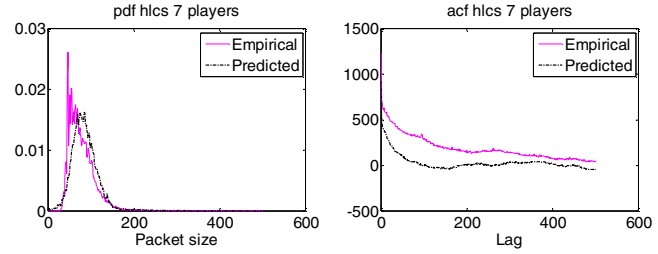

Fig. 7. Half-Life Counterstrike Probability Mass Function and Autocovariance Function for 7 Player Game

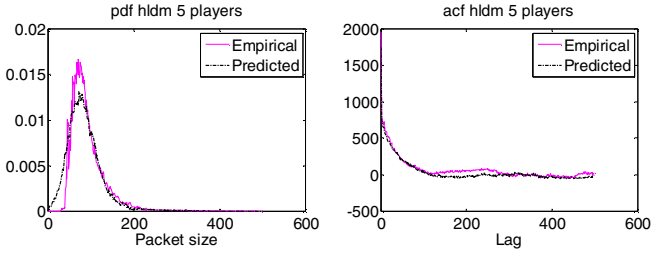

Fig. 8. Half-Life Death Match Probability Mass Function and Autocovariance Function for 5 Player Game

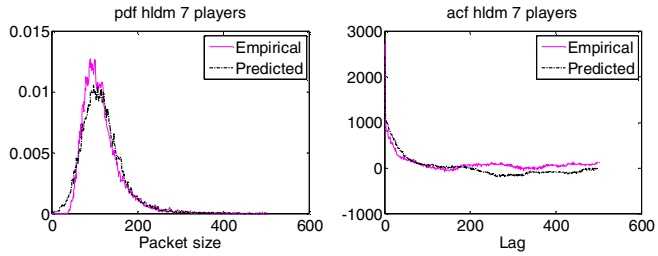

Fig. 9. Half-Life Death Match Probability Mass Function and Autocovariance Function for 7 Player Game

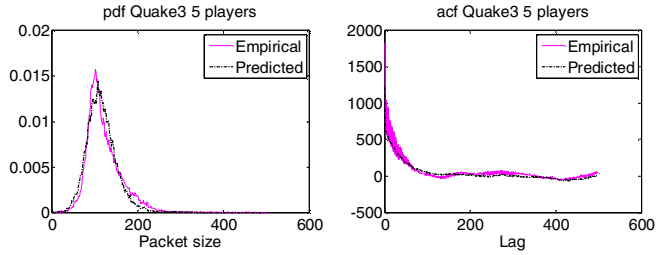

Fig. 10. Quake III Arena Probability Mass Function and Autocovariance Function for 5 Player Game

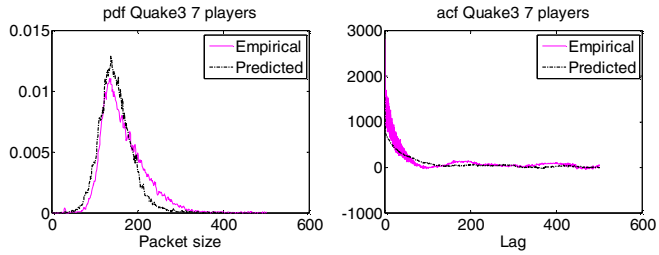

Fig. 11. Quake III Arena Probability Mass Function and Autocovariance Function for 7 Player Game

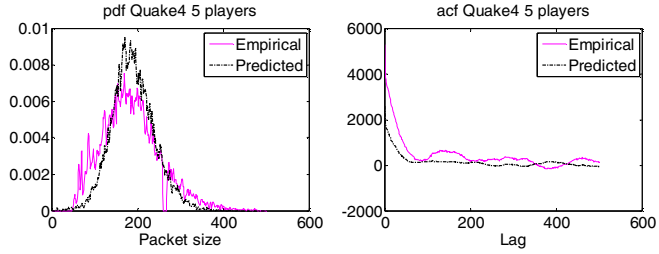

Fig. 12. Quake4 Probability Mass Function and Autocovariance Function for 5 Player Game

B. Analysis of Residuals

A consequence of our analysis is that the residuals of games with larger numbers of players should be able to be predicted from residuals of games with smaller numbers of players using Equations (7) and (8). In this section we illustrate that this is indeed the case. Fig. 13 shows the predicted and empirically obtained residuals. We see that there is an acceptable match. Once again, for reasons of space we present only a small number of examples to illustrate this point.

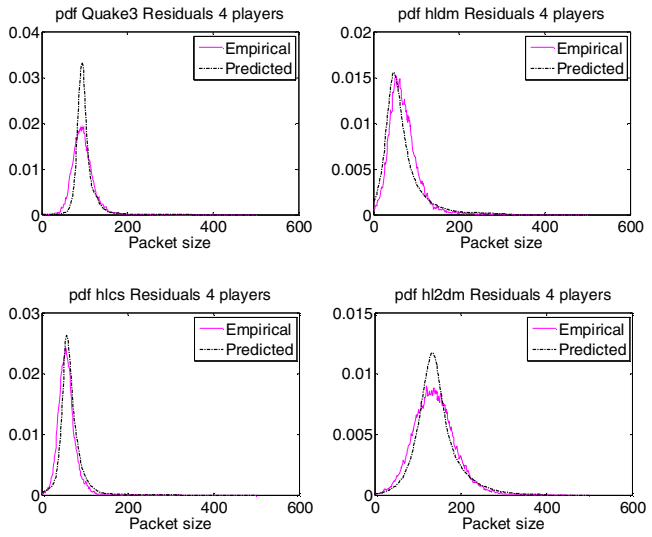

Fig. 13. Empirical and Predicted Residuals

C. *ARMA(1,1) parameters*

Equation (6) led us to predict that φ and θ should be constant for any game regardless of the number of players. Table 1 shows these values for a selection of games. We see that indeed, the values of φ and θ are very similar.

TABLE I
ARMA(1,1) Parameters

	HLDM		HLCS		Quake3	
Players	ϕ_1	θ_1	ϕ_1	θ_1	ϕ_1	θ_1
2	0.97	-0.84	0.96	-0.77	0.97	-0.77
3	0.97	-0.76	0.97	-0.77	0.98	-0.82
4	0.98	-0.80	0.98	-0.82	0.97	-0.81
5	0.97	-0.83	0.98	-0.81	0.98	-0.83
6	0.97	-0.81	0.98	-0.83	0.98	-0.84
7	0.96	-0.81	0.98	-0.81	0.98	-0.82
8	0.98	-0.85	0.98	-0.78	0.98	-0.83
9	0.97	-0.82	0.99	-0.82	0.98	-0.84

VI. CONCLUSION

In this paper we have presented evidence that suggests that FPS game traffic can be understood as being the aggregation of the largely independent behavior of multiple players. We have shown how an ARMA(1,1) model of a 2 and 3-player game can be extrapolated to games with larger numbers of players. This work should facilitate the construction of FPS game traffic simulators for new games.

In obtaining these models we have made simplifying assumptions that seem to be supported by the agreement between the empirical and synthetic models. Nevertheless, future research will involve investigation into the limits of the assumptions used to generate these models and how they may be need to be modified for new games.

FPS games, although popular and demanding of network resources, are not the only online games. Future research will involve application of the techniques described in this paper to other game styles.

Finally the whole purpose of developing models is to use them to investigate network performance issues. We will use these models to investigate how FPS games and other applications interact and what techniques are likely to be successful in minimizing their impact on each other.

REFERENCES

[1] M. Borella, "Source models of network game traffic," Computer Communications, vol. 23, pp. 403-410, Feb 2000.

[2] P. Branch and G. Armitage, "Extrapolating server to client IP traffic from empirical measurements of first person shooter games," in 5th Workshop on Network System Support for Games 2006 (Netgames2006) Singapore, 2006.

[3] P. Branch and G. Armitage, "Measuring the auto-correlation of server to client traffic in first person shooter games," in Australian Telecommunications, Network and Applications Conference (ATNAC) Melbourne, Australia, 2006.

[4] C. Chambers, W.-C. Feng, S. Sahu, and D. Saha, "Measurement-based characterization of a collection of on-line games," in Internet Measurement Conference 2005 (IMC2005) Berkeley California, 2005.

[5] J. Farber, "Traffic modelling for fast action network games," Multimedia Tools and Applications, vol. 23, pp. 31-46, Dec 22 2004.

[6] W.-C. Feng, F. Chang, W.-C. Feng, and J. Walpole, "A traffic charaterization of popular on-line games," IEEE/ACM Transactions on Networking (TON), vol. 13, pp. 488-500, June 2005.

[7] W.-C. Feng, F. Chang, W.-C. Feng, and J. Walpole, "Provisioning on-line games: A traffic analysis of a busy Counter-Strike server," in SIGCOMM Internet Measurement Workshop, Marseille, France, 2002.

[8] T. Henderson and S. Bhatti, "Modelling user behaviour in networked games," in 9th ACM International Conference on Multimedia (ACM Multimedia) Ottawa, Canada, 2001.

[9] T. Henderson and S. Bhatti, "Networked games - a QoS-sensitive application for QoS insensitive users?," in ACM SIGCOMM workshop on Revisiting IP QoS Karlsruhe, Germany, 2003.

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

[11] T. Lang, G. Armitage, P. Branch, and H.-Y. Choo, "A synthetic traffic model for Half-Life," in Australian Telecommunications, Networks and Applications Conference (ATNAC) Melbourne, 2003.

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

[13] S. Zander and G. Armitage, "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 (NOSSDAV2005) Washington, 2005.

[14] Centre for Advanced Internet Architectures, "Simulating online network games (SONG)," Accessed 14 March 2007, http://caia.swin.edu.au/sitcrc

[15] G. Armitage, M. Claypool, and P. Branch, "Networking and online games: Understanding and engineering multiplayer Internet games," Chichester, England: John Wiley and Sons Ltd, 2006.

[16] S. Floyd and E. Kohler, "Internet research needs better models," Computer Communications Review (CCR), vol. 33, pp. 29-34, January 2003.

[17] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW client-based traces," Boston University Computer Science Technical Report, vol. TR-96-08, 1995.

[18] P. Branch, A. Cricenti, and G. Armitage, "Modeling Server to Client IP traffic in First Person Shooter Games," in IEEE International Conference on Communications (ICC08) Beijing, 2008.

[19] A. Cricenti, P. Branch, and G. Armitage, "Time-series modelling of server to client IP packet length in first person shooter games," in International Conference on Networks 2007 (ICON07) Adelaide, 2007.

[20] H. Akaike, "A new look at statistical model identification," IEEE Transactions on Automatic Control, vol. 19, pp. 716-713, 1974.