ARMA(1,1) Modeling of Quake4 Server to Client Game Traffic

Anthony Cricenti
Centre for Advanced Internet Architecture
Swinburne University of Technology
Melbourne, Australia
+61 3 9214 5000
acricentia@swin.edu.au

Philip Branch
Centre for Advanced Internet Architecture
Swinburne University of Technology
Melbourne, Australia
+61 3 9214 5000
pbranch@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 autocorrelation models are important elements in the creation of realistic traffic generators for network simulators such as ns-2 and OMNET++. In this paper we show that ARMA(1,1) models capture the time series behaviour of Quake4 game traffic well. We also show that the random component of the ARMA models (the innovations) have distributions that appear to change little as the number of players increases.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *Client/server, Distributed applications.*

General Terms
Measurement, Performance, Theory.

Keywords
First Person Shooter, Online Games, Teletraffic Analysis, Traffic Engineering, Simulation, UDP, Traffic Modeling.

1. 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 so 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.

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 [15]. 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. A weakness of much of this modeling is that it has failed to capture the time series nature of game traffic. Most models have implicitly assumed that game traffic, while satisfactorily modeled by a particular distribution, is completely uncorrelated. That is, there is no relation between one packet size and the next.

In this paper we show that Quake4 FPS game traffic is correlated, that the correlation can be well captured using an ARMA(1,1) model, and very significantly, the distribution of the random variations incorporated in the ARMA(1,1) model (the innovations) have some surprising similarities regardless of the number of players.

To carry out this work, we used data gathered from game trials ran within the Centre for Advanced Internet Architectures during 2006 [4]

The paper is structured as follows. Section 2 reviews the basic network architecture and traffic patterns of modern FPS games. Section 3 is a brief introduction to time-series modeling, particularly ARMA models. Section 4 demonstrates the effectiveness of ARMA(1,1) in modeling the time-series behavior of Quake4 games. Section 5 presents probability density functions of the innovations for the Quake4 models. Section 6 discusses future research and concludes the paper.

2. FIRST PERSON SHOOTER GAMES
2.1 Client-Server Architecture
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. In principle clients might be designed to communicate directly with each other in a peer-to-peer fashion. However, in practice, 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

NetGames'07, September 19-20, 2007, Melbourne, Australia.

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.

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. (Games can 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 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 propagated to other players.

Game-state updates must occur in a timely and prompt manner, with minimal bias or favor towards any particular player. In FPS games, 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 with the default configuration sends bursts of four back-to-back update packets every 50ms, one IP packet to each active client [15]). Each client receives 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 40ms [15].

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 [1, 7].

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, 7].

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.

2.3 Phases of Gameplay and Game Traffic

In most FPS games there are three different 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.

3. AR(1) AND ARMA(1,1) MODELS

Mixed autoregressive / moving average (ARMA) models have been successfully used to model Variable Bit Rate video and ATM traffic [16]. We now show that they can be used to model the behaviour of Quake4 network traffic as well.

Intuitively an ARMA(1,1) model captures the correlated nature of a time varying signal through a combination of an autoregressive (AR) component and a moving average (MA) component. The parameter (1,1) describes the number of terms in the AR and MA components respectively.

For a Stationary time-series X_t, we define an ARMA(p,q) process as:

$$\phi(B) X_t = \theta(B) Z_t \quad (1)$$

where $\phi(B)$ is the autoregressive polynomial of degree p and $\theta(B)$ is the moving average polynomial of order q

$$\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p \quad (2)$$

$$\theta(B) = 1 + \theta_1 B + \ldots + \theta_q B^q \quad (3)$$

and B is the backshift operator defined as:

$$B^i X_t = X_{t-i} \quad i = 0, \pm 1, \pm 2... \tag{4}$$

The innovations Z_t in (1) are assumed to be independent identically-distributed (iid) random variables with zero mean and variance σ^2.

If $\theta(z) = 1$ we have a pure autoregressive (AR) process, while if $\phi(z) = 1$ we have a pure moving average process (MA)

Traditional Box-Jenkins ARMA models describe stationary time-series [17]. Intuitively a time-series is stationary if the statistical properties such as the mean and the variance are not time dependent. Generally if the values of the time-series fluctuate about a constant mean value without a trend then the time-series is stationary.

If the process is purely Markov than it can be modelled using an AR(1) model.

An AR(1) process is a special case of the ARMA(p,q) process with $\theta(z) = 1$ and $p = 1$. The model is given by the equation:

$$X_t = \phi_1 X_{t-1} + Z_t \tag{5}$$

Applying this to our packet length data (5) states that the current server to client packet payload size X_t is related to the size of the last packet sent X_{t-1} and an error term Z_t (the innovations).

4. ARMA(1,1) AND AR(1) MODELS OF QUAKE4 GAME TRAFFIC

In this section we show that an ARMA(1,1) process is successful in modelling the server to client packet size distribution of Quake4 games more successfully than the simpler AR(1) process.

Various ARMA models were fitted to the empirical data to determine whether the synthetic time-series model could better match the empirical data than does the simpler AR(1) model. The simplest ARMA model that can be fitted is the ARMA(1,1). Results for the AR(1) and ARMA(1,1) models are presented in Tables 1 to 4.

Using the ARMA(1,1) models a probability distribution of packet lengths can be obtained. These are shown (dotted lines) with the empirical data in fig. 1. From the plots it appears that the ARMA(1,1) while introducing some smoothing, is successful in predicting the probability distribution of the packet lengths.

In Tables 1 and 2 we compare statistics derived from simulations obtained from AR(1) and ARMA(1,l) models fitted to the empirical data. The variances of the data obtained from the AR(1) models are much poorer matches to the empirically derived variances than the variance of the ARMA(1,1) models. We can interpret this as suggesting that ARMA(1,1) models capture the autocorrelated nature of Quake4 traffic more effectively than a simple AR(1) model.

Table 1. Variance of AR(1) model for Quake4

Players	Empirical	Synthetic	Innovations
2	1163	1867	816
3	2587	3624	2025
4	3559	5346	2653
5	7155	9153	6095
6	9095	11400	7820
7	11124	15022	9150

Table 2. Variance of ARMA(1,1) model for Quake4.

Players	Empirical	Synthetic	Innovations
2	1163	1181	674
3	2587	2681	1647
4	3559	3506	2171
5	7155	6593	4909
6	9095	8735	6415
7	11124	10849	7497

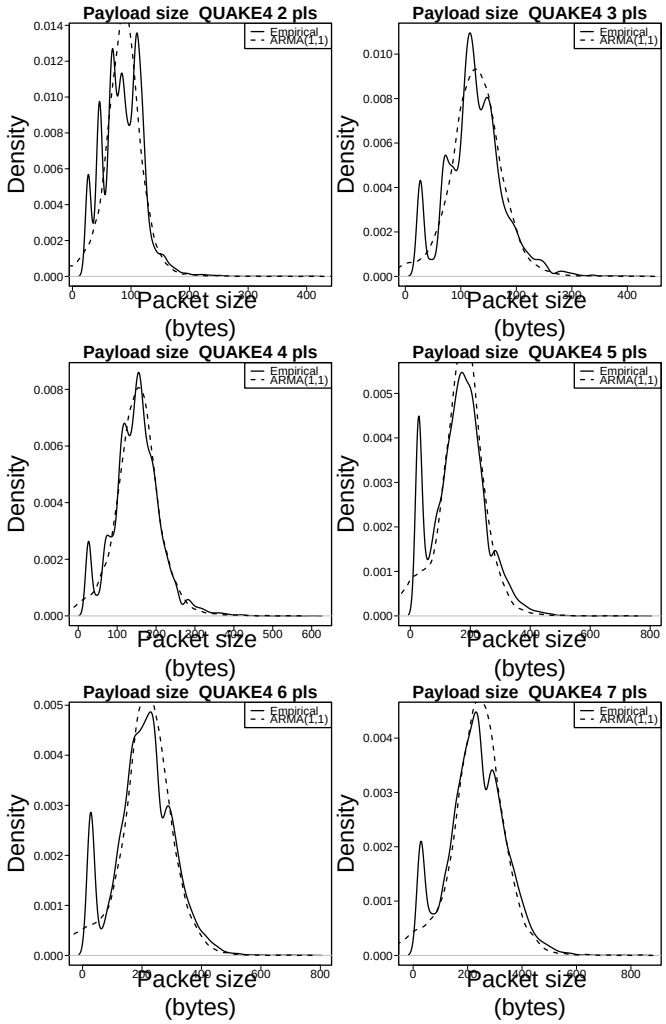

Fig. 1. Empirical and Synthetic Quake4 Packet Size PDFs

5. ARMA(1,1) INNOVATIONS OF THE QUAKE4 TRAFFIC MODELS

We now examine the innovations of the ARMA processes used to model the traffic.

In any ARMA process the random behaviour of the process is characterised by its 'innovations'. That is, each term in the sequence, while having a strong dependence on the previous value or values, deviates from the expected value in some random fashion. The sequence of these random values comprises the 'innovations' of the sequence.

Understanding the innovations enables us to predict the behaviour of the sequence. By knowing the probability distribution of the innovation sequence, an accurate representative sequence of samples can be generated.

We now present an analysis of the innovation distributions for 2 to 7 player games for Quake4. In all cases we see that the innovation sequence has very similar probability density functions. The only difference between them is their width, which is related to the variance of the sample dataset.

An important issue in understanding the nature of a timeseries is to identify the distribution of the innovations. Time series innovations are frequently modelled with Gaussian distributions or (less commonly) Laplace distribution. We now present Q-Q plots of the Quake4 2-player games comparing the empirically obtained distributions with theoretical distributions. A Q-Q plot is a qualitative measure of the effectiveness of a theoretical distribution in modelling an empirically obtained distribution. The sample quantiles are mapped against the theoretical quantiles. A good match will have all points of the mapping falling on a diagonal line. Clearly neither the Laplace or the Gaussian is a good model of the innovation distribution..

The Q-Q plots also show that the innovation distributions are not symmetric. Both the Laplace and Gaussian fail to capture the long-tail nature of the innovation distribution. While the ARMA(1,1) model appears a good model of Quake4 traffic, better understanding of the random component is needed. This is an area of future work.

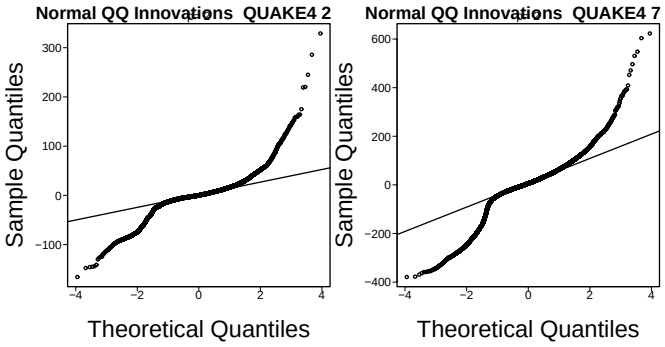

Fig. 2. Q-Q plots comparing the innovations of the ARMA(1,1) model to a Gaussian distribution 2 and 7 player games

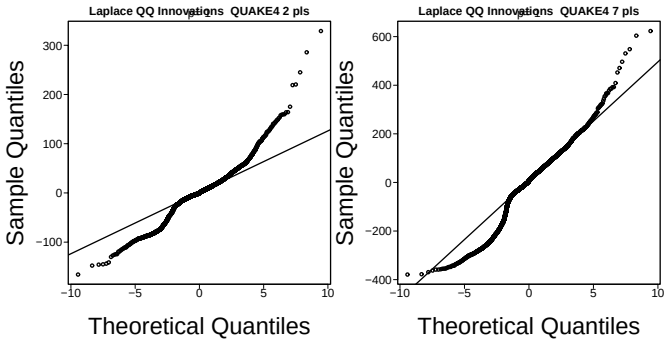

Fig. 3. Q-Q plots comparing the innovations of the ARMA(1,1) model to a Laplace distribution 2 and 7 player games

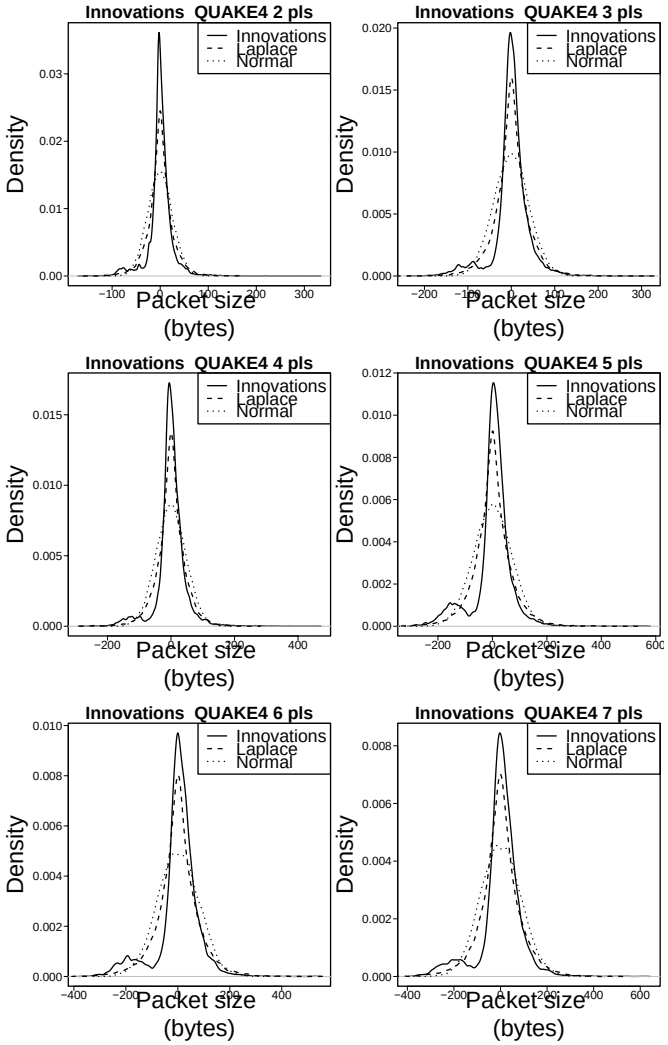

Fig. 4. Time series innovations of the ARMA(1,1) model for Quake4 2 to 7 player games

6. CONCLUSION AND FUTURE RESEARCH

This paper has shown that ARMA(1,1) is a good model of the time series behavior of Quake4 first person shooter game traffic. It has also presented an analysis of the innovations of the model and has demonstrated that there are some interesting commonalities across the games. All games analysed have very similar innovation probability density functions, differing only in their scaling.

Future research will attempt to apply these commonalities to developing models that can be extrapolated to larger number of players as well as implement simulations based on the ARMA(1,1) model. We will also investigate whether these techniques and commonalities hold with other FPS games as well as with different game genres.

Understanding the nature of innovations means that good simulations of game traffic can be produced. It also may provide some insights into why game traffic behaves as it does. An interesting question for future research is to attempt to understand why (apart from scaling) the innovations are so similar across all the different games and numbers of players and to identify some way of characterising the innovation distribution.

Finally, future research will involve the development of simulation systems that implement these models and can be applied to solve significant network problems.

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

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

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

[16] R. Grunenfelder, J. Cosmas, S. Manthorpe, and A. Odinma-Okafor, "Characterization of video codecs as autoregressive moving average processes and related queueing system performance," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 284-293, April 1991.

[17] G. Box and G. Jenkins, *Time series analysis: Forecasting and control*, revised ed. San Francisco: Holden-Day Inc, 1976.