Traffic Modelling of On-line First Person Shooter Games

By
Antonio Luciano Cricenti

A thesis submitted for the degree of
Doctor of Philosophy

Faculty of Information and Communication Technologies,
Swinburne University of Technology.

June 2014
Abstract

Over the past few decades, there has been a great deal of attention paid to modelling the traffic generated by First-person Shooter (FPS) games. This interest has been motivated by the large growth in on-line gaming, which is now a multi-billion dollar industry. The FPS genre is of particular importance to the network engineering community due to its highly interactive nature. The interest in this genre is based not only on the effects of the network on playability and user experience of the game, but also on the effects that traffic produced by this genre of games may have on other traffic flowing in the network. In order that network simulation can provide more insightful and useful results for studying the interaction between this game traffic and other classes of traffic, good traffic models are required to be developed. This thesis makes a number of contributions to the state of the art in game traffic modelling and synthesis. We show that currently used models are incomplete in that they do not scale with the number of users, nor do they capture the autocorrelated nature of the traffic, and they do not take into account behaviour of individual FPS games. We address each of these matters. In particular, a general model for FPS game traffic during the game-play phase is presented. The development of this traffic model was based on measurements of traffic produced by seven popular FPS games. This model, which is based on the Ex-Gaussian and Gamma Modified Gaussian mixture distributions, is novel and computationally simple and it can be scaled to model scenarios with a large number of game participants without having to conduct and collect data from large numbers of controlled experiments. Since this is a computationally simple model, it is ideal for use as the basis of either a hardware or software based traffic generator that can be used to synthesise representative FPS game traffic. This model is an improvement over previous models, as it also incorporates the serial autocorrelation that exists in the packet traffic that flows in the server-to-client direction by using a simple ARMA time-series approach. A traffic generator based on this model has also been developed for use with the OMNet++ simulation software.
This traffic model can be used by Internet Service Providers (ISPs) and the network engineering community to investigate network performance issues, arising from scenarios where FPS games and other applications interact and what techniques are likely to be successful in minimizing their impact on each other.
Ackowledgments

I would like to thank all the people who have guided me in my life and helped me during this journey. In particular I must express my deep gratitude to my family (Rosie, Vincent and Marcus), whose emotional support through this period of my life has made it bearable. I would like to express my gratitude to my Principal Coordinating Supervisor, Dr. Philip Branch for passing on so much advice and of course for freely giving so much of his time. I appreciate his vast knowledge and skill in many areas and his assistance in writing papers and reports. My thanks go out to my Coordinating Supervisor Professor Grenville Armitage for his support during my candidature and his useful advice during conversations in the corridor.

A special thank-you goes out to my great friend and colleague Dragi Klimovski, for his support and for keeping me motivated when things got tough.

I must acknowledge the GENIUS project, which is the home of networked game research in the Centre for Advanced Internet Architectures (CAIA), as this has underpinned and provided the environment to nurture my work. I would also like to acknowledge the members of CAIA who were involved in creating and documenting the game traffic trace files, without their contribution, this work would not have been possible.

Finally, I would like to express my gratitude to the Faculty of Information & Communication Technologies at the Swinburne University of Technology for their support during my candidature.
Declaration

I declare that this thesis contains no material which has been accepted for the award of any other degree or diploma, except where due reference is made in the text of the thesis. To the best of my knowledge the thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Signed Date: 5th May 2014

A.L. Cricenti
List of Publications

Elements of the work presented in this thesis have already appeared in the following peer-reviewed publications authored in conjunction with my supervisors.


Contents

1 Introduction ................................................................. 1
  1.1 Background ........................................................... 1
  1.2 On-line Multi-player Games ........................................ 3
  1.3 Motivation ............................................................... 3
  1.4 Scope of Research .................................................... 4
  1.5 Thesis Outline .......................................................... 5

2 First-person Shooter Games Traffic .................................. 7
  2.1 Introduction ............................................................ 7
  2.2 Traffic Modelling ....................................................... 9
  2.3 First-person Shooter Games ......................................... 11
    2.3.1 FPS Game Traffic Source Models ............................. 11
    2.3.2 Traffic Compression ............................................ 13
  2.4 Phases of Game-play and Game Traffic ............................ 15
  2.5 Session Level ............................................................. 16
    2.5.1 Session membership ............................................ 16
    2.5.2 Session Inter-arrivals ......................................... 17
    2.5.3 Session Duration ................................................ 17
    2.5.4 Geographical Distribution of Servers and Players ............. 19
  2.6 Modelling the Start-up Phase ....................................... 20
    2.6.1 Server Discovery Process .................................... 21
    2.6.2 Server Discover Traffic ...................................... 21
  2.7 Modelling the Game-play Phase ..................................... 23
    2.7.1 Probability Density Function Based Traffic Modelling ...... 24
    2.7.2 Assumptions in First Person Shooter FPS Game Traffic Modelling ................................ 25
    2.7.3 Client-to-Server Inter-arrival time .......................... 27
2.7.4 Client-to-Server (C2S) Payload Size ........................................ 29
2.7.5 Server-to-Client Inter-arrival time (IAT) .................................. 29
2.7.6 Server-to-Client Payload Size. .................................................. 31
2.7.7 Summary ................................................................. 34
2.8 Extrapolation of models to N-players ............................................. 34
  2.8.1 Existing Extrapolation Methods ............................................. 35
2.9 Correlation Effects in FPS Traffic ................................................ 36
  2.9.1 Time-Series Modelling of FPS Game Traffic ............................... 37
  2.9.2 Background on Self-similarity .............................................. 38
2.10 Conclusion ................................................................. 38

3 Preliminaries and Methodology ....................................................... 41
  3.1 Introduction ........................................................................... 41
  3.2 Tools for Stochastic Processes .................................................... 42
    3.2.1 Distribution Function ...................................................... 42
    3.2.2 Density Function .......................................................... 42
    3.2.3 Sum of Random Variables ................................................ 42
    3.2.4 Characteristic Function ................................................... 43
  3.3 Standard Distributions ............................................................. 45
    3.3.1 Degenerate Distribution .................................................... 45
    3.3.2 Gaussian Distribution ..................................................... 46
    3.3.3 Log-Normal Distribution .................................................. 47
    3.3.4 Generalised Extreme Value Distribution ............................... 48
    3.3.5 Gamma Distribution ......................................................... 49
  3.4 Mixture Distributions .............................................................. 50
    3.4.1 Ex-Gaussian Distribution .................................................. 51
    3.4.2 Gamma Modified Gaussian Distribution ................................ 52
    3.4.3 Linear combination of Gamma Modified Gaussian distributed
         variables ........................................................................ 53
  3.5 Parameter Estimation ............................................................. 55
    3.5.1 Method of Moments ......................................................... 56
    3.5.2 Maximum Likelihood Estimation .......................................... 56
  3.6 Quantile-Quantile Plots ........................................................... 57
  3.7 Model Discrepancy ............................................................... 58
  3.8 Autocovariance and Autocorrelation Functions ............................. 59
  3.9 Stationarity ............................................................................. 60
CONTENTS

3.10 Time-series Models .................................................. 61
  3.10.1 Parsimony ....................................................... 62
3.11 Generalized Autoregressive Moving Average Model ............... 62
  3.11.1 Short-Range Dependent (SRD) Processes .................... 64
  3.11.2 Autoregressive Integrated Moving Average Process .......... 66
  3.11.3 Long-Range Dependent (LRD) Time-series Models .......... 66
3.12 Collection of Traffic Traces ...................................... 67
3.13 Summary ............................................................ 69

4 FPS Games Traffic Traces ............................................. 71
  4.1 Introduction ....................................................... 71
  4.2 First-person Shooter Traffic .................................... 71
    4.2.1 Traffic Traces ............................................... 72
    4.2.2 Truncating the Traffic Traces .............................. 72
  4.3 Packet Inter-arrival Times ...................................... 73
  4.4 Payload Size ..................................................... 75
    4.4.1 Descriptive Statistics - Quake III Arena ................. 75
    4.4.2 Descriptive Statistics - Wolfenstein Enemy Territory Pro ... 79
    4.4.3 Descriptive Statistics Half-Life 2 ......................... 84
    4.4.4 Descriptive Statistics - Counter-Strike-Source .......... 87
    4.4.5 Descriptive Statistics - Half-Life ......................... 91
    4.4.6 Descriptive Statistics - Half-Life Counter-Strike ....... 94
    4.4.7 Descriptive Statistics - Quake IV ......................... 98
  4.5 Discussion ...................................................... 102
  4.6 Summary .......................................................... 103

5 Standard Distribution Models ....................................... 105
  5.1 Introduction ..................................................... 105
  5.2 Standard Distribution Based Models for Payloads of FPS Games .. 106
  5.3 Distribution Fitting for Payloads of FPS Games ................ 106
    5.3.1 Client-to-Server Payload Size ............................ 107
    5.3.2 Server-to-Client Payload ................................... 114
    5.3.3 Discussion of distribution fitting to the server-to-client payload 121
  5.4 Summary .......................................................... 123

6 Time Series Modelling ................................................. 125
  6.1 Introduction ..................................................... 125
## CONTENTS

6.2 Time-series Modelling of FPS Games .................................. 126
  6.2.1 Models for the Server-to-Client (S2C) Payload for FPS games 126
  6.2.2 Server-to-Client First Order AR(1) Models ...................... 127
  6.2.3 Server-to-Client AR(1) Models ................................. 128
  6.2.4 Higher order AR(p) Models Server-to-Client ................. 132
  6.2.5 Server-to-Client ARMA(p,q) Models ............................ 140

6.3 Conclusions .......................................................... 148

7 N-player Extrapolation Models ........................................... 151
  7.1 Introduction ....................................................... 151
  7.2 Extrapolating Standard Distribution Models ...................... 152
  7.3 Extrapolation using the Ex-Gaussian (EG) distribution ........... 154
  7.4 Results from the EG extrapolation method ......................... 157
  7.5 Extrapolating Time-series models ................................ 160
    7.5.1 Extrapolation of GARMA(p,q,δ,η) models to N-player games 161
    7.5.2 Repeated Convolution Extrapolation Technique ............... 161
    7.5.3 ‘Per player’ Residual Density ............................... 163
    7.5.4 ARMA Models ................................................. 164
  7.6 FPS ARMA extrapolation models .................................... 165
    7.6.1 Server-to-Client Payload Residual Density ................. 166
    7.6.2 Server-to-Client Payload Length Density and ACVF .......... 168
  7.7 ‘Per Player’ Residual Density .................................... 176
  7.8 Extrapolating the Residual Density ................................ 180
    7.8.1 The Variance Gamma Modified Gaussian Density as the Re-
    sidual Density ................................................ 180
  7.9 Conclusion ........................................................ 182

8 Simulation Models ........................................................ 189
  8.1 Introduction ....................................................... 189
  8.2 OMNeT++ based FPS Game Traffic Generator ....................... 189
    8.2.1 General Traffic Generator Details .......................... 190
  8.3 Traffic Generator Validation ...................................... 190
  8.4 IEEE802.16 Model vs ΓMG Model .................................. 193
  8.5 Summary .......................................................... 194

9 Conclusions ............................................................. 197
  9.1 Summary .......................................................... 197
9.2 Contributions .................................................... 200
9.3 Future research .................................................. 201

A Glossary .............................................................. 219

B OMNeT++ code ...................................................... 223
  B.1 Model Parameters ............................................... 223
      B.1.1 GameBasicApp Parameters ............................. 223
      B.1.2 ExGaussArmaApp Parameters ......................... 224
  B.2 GameBasicApp.ned ............................................. 227
  B.3 GameBasicApp.h ............................................... 227
  B.4 GameBasicApp.cc .............................................. 227
  B.5 ExGaussArmaApp .ned ........................................ 227
  B.6 ExGaussArmaApp .h ............................................ 227
  B.7 ExGaussArmaApp .cc .......................................... 227

C Gamma Modified Gaussian Distribution ...................... 229
  C.1 Density of the Gamma Modified Gaussian (ΓMG) Distribution . . . . 229
  C.2 Density of the Ex-Gaussian Distribution .................... 232
List of Figures

2.1 FPS Game Scenario .................................................. 12
2.2 Average great-circle distance of players over time. From [55] .... 20
2.3 Wolfenstein Enemy Territory Pro (ETPro) client’s server discovery
    process. From [5] ......................................................... 21
2.4 Median Inter-probe interval vs time of day. From [5] .............. 22
2.5 FPS Game Traffic ..................................................... 24

3.1 Typical PDFs of various distributions .................................. 46
3.2 Typical Quantile-Quantile Plot (Q-Q) plots for empirical data vs dis-
    tribution ................................................................. 58
3.3 Traffic Capture Set-up ................................................. 68

4.1 ETPro (a) Complete Packet Trace. (b) Truncated Packet Trace ... 73
4.2 Quake III Arena: S2C Mean, Median and Variance vs Number of
    Players ................................................................. 77
4.3 Quake III Arena: Density of S2C Payload Size ....................... 77
4.4 Quake III Arena: Spectrum S2C Payload Size ....................... 78
4.5 Quake III Arena: ACVF S2C Payload Size: Truncated Trace ........ 78
4.6 Quake III Arena: ACVF C2S Payload Size: Truncated Trace ........ 79
4.7 Wolfenstein Enemy Territory Pro: S2C Mean, Median and Variance
    vs Number of Players ................................................ 81
4.8 Wolfenstein Enemy Territory Pro: Density of S2C Payload Size ... 82
4.9 Wolfenstein Enemy Territory Pro: Spectrum S2C Raw Payload Size . 82
4.10 Wolfenstein Enemy Territory Pro: S2C Payload Size: Truncated Raw
    Trace ................................................................. 82
4.11 Wolfenstein Enemy Territory Pro: Density, Spectrum S2C Payload
    Size: De-seasonalised Trace ....................................... 83
LIST OF FIGURES

4.12 Wolfenstein Enemy Territory Pro: Autocovariance Function (ACVF) S2C Payload Size: De-seasonalised Trace ......................... 83
4.13 Wolfenstein Enemy Territory Pro: ACVF C2S Payload Size: Truncated Trace ....................................................... 84
4.14 Half-Life 2: S2C Mean and Median vs Number of Players ......... 85
4.15 Half-Life 2: Density of S2C Payload Size ............................. 86
4.16 Half-Life 2: Spectrum S2C Payload Size .............................. 86
4.17 Half-Life 2: ACVF S2C Payload Size: Raw Trace .................. 87
4.18 Half-Life 2: ACVF C2S Payload Size: Truncated Trace .......... 87
4.19 Counter-Strike-Source: S2C Mean, Median and Variance vs Number of Players ......................................................... 89
4.20 Counter-Strike-Source: Density of S2C Payload Size ............... 89
4.21 Counter-Strike-Source: Spectrum S2C Payload Size ............... 89
4.22 Counter-Strike-Source: ACVF S2C Payload Size: Truncated Trace . 90
4.23 Counter-Strike-Source: ACVF C2S Payload Size: Truncated Trace . 91
4.24 Half-Life: S2C Mean, Median and Variance vs Number of Players .. 92
4.25 Half-Life: Density of S2C Payload Size .............................. 92
4.26 Half-Life: Spectrum S2C Payload Size ............................... 93
4.27 Half-Life: ACVF S2C Payload Size: Truncated Trace ............. 93
4.28 Half-Life: ACVF C2S Payload Size: Truncated Trace ............ 94
4.29 Half-Life Counter-Strike: S2C Mean, Median and Variance vs Number of Players ......................................................... 95
4.30 Half-Life Counter-Strike: Density of S2C Payload Size .......... 96
4.31 Half-Life Counter-Strike: Spectrum S2C Payload Size ............ 96
4.32 Half-Life Counter-Strike: ACVF S2C Payload Size: Truncated Trace . 97
4.33 Half-Life Counter-Strike: ACVF C2S Payload Size: Truncated Trace . 97
4.34 Quake IV: S2C Mean, Median and Variance vs Number of Players . 99
4.35 Quake IV: Density of S2C Payload Size ................................ 99
4.36 Quake IV: Spectrum S2C Payload Size .............................. 100
4.37 Quake IV: Spectrum S2C Payload Size: De-seasonalised Trace ... 100
4.38 Quake IV: ACVF S2C Payload Size: Raw Trace .................... 100
4.39 Quake IV: ACVF S2C Payload Size:: De-seasonalised Trace ...... 101
4.40 Quake IV: ACVF C2S Payload Size: Truncated Trace ............ 101
5.1 Client-to-Server Payload Size Density and Distribution for Wolfenstein Enemy Territory Pro ................................. 109
5.2 Client-to-Server Payload Size Q-Q plots for Wolfenstein Enemy Territory Pro (Sanitised Data) .................................................. 109
5.3 Client-to-Server Payload Size Density and Distribution for Counter-Strike-Source ................................................................. 109
5.4 Client-to-Server Payload Size Density and Distribution for Half-Life 2 110
5.5 Client-to-Server Payload Size Q-Q plots for Half-Life 2 (Sanitised Data) 110
5.6 Client-to-Server Payload Size Density and Distribution for Half-Life Counter-Strike ................................................................. 111
5.7 Client-to-Server Payload Size Q-Q plots for Half-Life Counter-Strike (Sanitised Data) ................................................................. 112
5.8 Client-to-Server Payload Size Density and Distribution for Half-Life) 112
5.9 Client-to-Server Payload Size Density and Distribution for Quake III Arena .......................................................... 113
5.10 Client-to-Server Payload Size Density and Distribution for Quake IV 113
5.11 Server-to-Client Payload Size Density and Distribution for Wolfenstein Enemy Territory Pro .................................................. 115
5.12 Server-to-Client Payload Size Q-Q plots for Wolfenstein Enemy Territory Pro (Sanitised Data) .................................................. 116
5.13 Server-to-Client Payload Size Density and Distribution for Counter-Strike-Source ................................................................. 117
5.14 Server-to-Client Payload Size Density and Distribution for Counter-Strike-Source (Sanitised Data) .................................................. 117
5.15 Server-to-Client Payload Size Density and Distribution for Half-Life 2 118
5.16 Server-to-Client Payload Size Density and Distribution for Half-Life Counter-Strike ................................................................. 118
5.17 Server-to-Client Payload Size Density and Distribution for Half-Life Counter-Strike (Sanitised Data) .................................................. 119
5.18 Server-to-Client Payload Size Q-Q plots for Half-Life Counter-Strike (Aggressively Sanitised Data) .................................................. 119
5.19 Server-to-Client Payload Size Density and Distribution for Half-Life . 120
5.20 Server-to-Client Payload Size Density and Distribution for Quake III Arena .......................................................... 121
5.21 Server-to-Client Payload Size Density and Distribution for Quake IV (Sanitised Data) ................................................................. 121

6.1 AR() model driven by the residuals sequence .............................................. 128
x

LIST OF FIGURES
6.2

Quake III Arena: Typical Density of S2C Payload Size: AR(1) . . . . 130

6.3

Half-Life Counter-Strike: Density of S2C Payload Size: AR(1) . . . . 130

6.4

Half-Life: Density of S2C Payload Size: AR(1) . . . . . . . . . . . . . 130

6.5

Counter-Strike-Source: Density of S2C Payload Size: AR(1) . . . . . 131

6.6

Half-Life 2: Density of S2C Payload Size: AR(1) . . . . . . . . . . . . 131

6.7

ETPro: Density of S2C Payload Size: AR(1) . . . . . . . . . . . . . . 131

6.8

Quake IV (Quake4): Density of S2C Payload Size: AR(1) . . . . . . . 132

6.9

Quake4: Density of S2C Payload Size: AR(1) Sanitised . . . . . . . . 133

6.10 Quake III Arena (Quake3): Normalised Variance and AIC of S2C
Payload Size: AR(p) . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
6.11 Half-Life Counter-Strike: Normalised Variance and AIC of S2C Payload Size: AR(p) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
6.12 Half-Life: Normalised Variance and AIC of S2C Payload Size: AR(p) 135
6.13 Counter-Strike-Source: Normalised Variance and AIC of S2C Payload
Size: AR(p) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
6.14 Half-Life 2: Normalised Variance and AIC of S2C Payload Size: AR(p)135
6.15 Wolfenstein Enemy Territory Pro: Normalised Variance and AIC of
S2C Payload Size: AR(p)

. . . . . . . . . . . . . . . . . . . . . . . . 136

6.16 Quake III Arena: Density of S2C Payload Size: AR(p) . . . . . . . . 136
6.17 Half-Life Counter-Strike: Density of S2C Payload Size: AR(p) . . . . 136
6.18 Half-Life: Density of S2C Payload Size: AR(p) . . . . . . . . . . . . . 137
6.19 Counter-Strike-Source: Density of S2C Payload Size: AR(p) . . . . . 137
6.20 Half-Life 2: Density of S2C Payload Size: AR(p) . . . . . . . . . . . . 137
6.21 Wolfenstein Enemy Territory Pro: Density of S2C Payload Size: AR(p)138
6.22 Quake IV: Normalised Variance and AIC of S2C Payload Size: AR(p)
Clean . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 139
6.23 Quake IV: Density of Sanitised S2C Payload Size: AR(p) Clean . . . 139
6.24 Quake IV: Truncating the Density Removing -ve Packet Lengths . . . 140
6.25 Quake III Arena: Density of S2C Payload Size: ARMA(1,1) . . . . . 142
6.26 ETPro: Density of S2C Payload Size: ARMA(1,1) . . . . . . . . . . . 143
6.27 Half-Life Counter-Strike: Density of S2C Payload Size: ARMA(1,1) . 143
6.28 Half-Life: Density of S2C Payload Size: ARMA(1,1) . . . . . . . . . . 143
6.29 Counter-Strike-Source: Density of S2C Payload Size: ARMA(1,1) . . 144
6.30 Half-Life 2: Density of S2C Payload Size: ARMA(1,1) . . . . . . . . . 144
6.31 Quake III Arena: Normalised Variance and AIC of S2C Payload Size:
ARMA() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145


LIST OF FIGURES

6.32 Wolfenstein Enemy Territory Pro: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1) ............................................. 145
6.33 Half-Life Counter-Strike (HLCS): Normalised Variance and AIC of S2C Payload Size: ARMA(p,1) ............................................. 145
6.34 Half-Life: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1) ................................................................. 146
6.35 Counter-Strike-Source: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1) ................................................................. 146
6.36 Half-Life 2: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1) ................................................................. 146
6.37 Quake IV: Density of S2C Payload Size: ARMA(1,1) ..................... 147
6.38 Quake IV: Density of S2C Payload Size: ARMA(1,1) (Sanitised Data) 148
6.39 Quake4: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1) 148
6.40 Quake IV: Density of S2C Payload Size: ARMA(1,1) ..................... 149

7.1 ETPro: Empirical and Synthetic CDFs ........................................ 158
7.2 HL2CS: Empirical and Synthetic CDFs ........................................ 158
7.3 HL2DM: Empirical and Synthetic CDFs ........................................ 159
7.4 HLCS: Empirical and Synthetic CDFs ........................................... 159
7.5 HLDM: Empirical and Synthetic CDFs ........................................... 159
7.6 Quake3: Empirical and Synthetic CDFs ........................................ 160
7.7 Quake4: Empirical and Synthetic CDFs ........................................ 160
7.8 ‘Per player’ Residual Density ..................................................... 167
7.9 ETPro: S2C ARMA(1,1) Residuals Density .................................... 168
7.10 Half-Life (HLDM): S2C ARMA(1,1) Residuals Density .................... 169
7.11 Half-Life 2 (HL2DM): S2C ARMA(1,1) Residuals Density ............... 169
7.12 HLCS: S2C ARMA(1,1) Residuals Density .................................... 169
7.13 Quake4: S2C ARMA(1,1) Residuals Density .................................... 170
7.14 Quake3: S2C ARMA(1,1) Residuals Density .................................... 170
7.15 Counter-Strike-Source (HL2CS): S2C ARMA(1,1) Residuals Density 170
7.16 Wolfenstein Enemy Territory Pro: Density of S2C Payload Size .......... 172
7.17 Half-Life 2: Density of S2C Payload Size ..................................... 172
7.18 Quake III Arena: Density of S2C Payload Size ............................. 172
7.19 Quake IV: Density of S2C Payload Size ....................................... 173
7.20 Half-Life: Density of S2C Payload Size ....................................... 173
7.21 Half-Life Counter-Strike: Density of S2C Payload Size ................... 173
7.22 Counter-Strike-Source: Density of S2C Payload Size ..................... 174
7.23 Wolfenstein Enemy Territory Pro: ACVF of S2C Payload Size . . . . 174
7.24 Quake III Arena: ACVF of S2C Payload Size ............................. 174
7.25 Quake IV: ACVF of S2C Payload Size ............................ 175
7.26 Half-Life: ACVF of S2C Payload Size ........................................ 175
7.27 Half-Life 2: ACVF of S2C Payload Size ................................. 175
7.28 Counter-Strike-Source: ACVF of S2C Payload Size .................. 176
7.29 Half-Life Counter-Strike: ACVF of S2C Payload Size ................ 176
7.30 Wolfenstein Enemy Territory Pro: S2C Packet Payload Size: ARMA(1,1) Extrapolated .......................... 181
7.31 Counter-Strike-Source: S2C Packet Payload Size: ARMA(1,1) Extrapolated .................................................. 182
7.32 Half-Life 2: S2C Packet Payload Size: ARMA(1,1) Extrapolated ... 183
7.33 Half-Life Counter-Strike: S2C Packet Payload Size: ARMA(1,1) Extrapolated ........................................................... 184
7.34 Half-Life: S2C Packet Payload Size: ARMA(1,1) Extrapolated .... 185
7.35 Quake III Arena: S2C Packet Payload Size: ARMA(1,1) Extrapolated 186
7.36 Quake IV: S2C Packet Payload Size: ARMA(1,1) Extrapolated .... 187

8.1 CDF S2C Mean Packet Payload Size ........................................ 192
8.2 S2C Mean Packet Payload Size Extrapolation ............................. 193
8.3 OMNeT++ Simulation Scenario .............................................. 194
8.4 Mean Throughput ............................................................... 195
List of Tables

2.1 Client-to-Server Inter-arrival time ........................................... 28
2.2 Client-to-Server Payload Size .................................................. 30
2.3 Server-to-Client Inter-arrival time ............................................. 32
2.4 Server-to-Client Payload Size ................................................... 33
3.1 Time Stamping Accuracy [119] .................................................. 68
4.1 Inter-arrival time ................................................................. 75
4.2 Descriptive Statistics Quake III Arena: S2C Trace ......................... 77
4.3 Descriptive Statistics Quake III Arena C2S: Trace ......................... 79
4.4 Descriptive Statistics Wolfenstein Enemy Territory Pro: S2C Trace .... 81
4.5 Descriptive Statistics Wolfenstein Enemy Territory Pro: C2S Trace .... 84
4.6 Descriptive Statistics Half-Life 2: S2C Trace ............................... 85
4.7 Descriptive Statistics Half-Life 2: C2S Trace ............................... 87
4.8 Descriptive Statistics Counter-Strike-Source: S2C Trace ................. 88
4.9 Descriptive Statistics Counter-Strike-Source: C2S Trace ................. 90
4.10 Descriptive Statistics Half-Life: S2C Trace ................................. 92
4.11 Descriptive Statistics Half-Life: C2S Trace ................................. 93
4.12 Descriptive Statistics Half-Life Counter-Strike: S2C Trace .............. 95
4.13 Descriptive Statistics Half-Life Counter-Strike: C2S Trace .............. 97
4.14 Descriptive Statistics Quake IV: S2C Trace ................................. 98
4.15 Descriptive Statistics Quake IV: C2S Trace ................................. 101
4.16 Characteristics of S2C Traffic Traces ....................................... 102
5.1 Distribution Parameters for C2S Payload Length ........................... 107
5.2 Distribution Parameters for S2C Payload Length ........................... 107
5.3 Statistics for C2S Payload Size ............................................... 108
5.4 Best Fit Distribution for C2S Payload Size .................................. 114
5.5 Statistics for the S2C Payload Length ....................................... 115
<table>
<thead>
<tr>
<th>Table No.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>Best Fit Distribution for S2C Payload Size</td>
<td>122</td>
</tr>
<tr>
<td>6.1</td>
<td>S2C payload AR(1) $\phi_1$ Parameter</td>
<td>128</td>
</tr>
<tr>
<td>6.2</td>
<td>Variance for Empirical and Synthetic AR(1) S2C payload</td>
<td>129</td>
</tr>
<tr>
<td>6.3</td>
<td>Quake4: Variance for Empirical and Synthetic AR(1) S2C payload</td>
<td>132</td>
</tr>
<tr>
<td>6.4</td>
<td>S2C ARMA(1,1) $\phi_1$, $\theta_1$ Parameter</td>
<td>141</td>
</tr>
<tr>
<td>6.5</td>
<td>Variance for Empirical and Synthetic ARMA(1,1) S2C payload</td>
<td>141</td>
</tr>
<tr>
<td>6.6</td>
<td>AIC for AR(1) and ARMA(1,1) models</td>
<td>141</td>
</tr>
<tr>
<td>6.7</td>
<td>Quake IV Variance for Empirical and Synthetic ARMA(1,1) S2C payload size</td>
<td>147</td>
</tr>
<tr>
<td>7.1</td>
<td>Timings: Repeated Convolutions vs Variance Gamma Modified Gaussian (VTG)</td>
<td>180</td>
</tr>
<tr>
<td>8.1</td>
<td>Distribution Parameters for S2C Packet Payload Length</td>
<td>191</td>
</tr>
<tr>
<td>8.2</td>
<td>Simulation Scenario Parameters</td>
<td>194</td>
</tr>
<tr>
<td>B.1</td>
<td>Parameters for the .ini file</td>
<td>224</td>
</tr>
<tr>
<td>B.2</td>
<td>Parameters for the Game Basic Application</td>
<td>224</td>
</tr>
<tr>
<td>B.3</td>
<td>Methods for the Game Basic Application</td>
<td>225</td>
</tr>
<tr>
<td>B.4</td>
<td>Methods for the ExGaussArmaApp Application</td>
<td>226</td>
</tr>
<tr>
<td>B.5</td>
<td>Parameters for the ExGaussArmaApp</td>
<td>226</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

The Internet, which is a global information systems network, is of great economic and social importance to today’s society. With the advent of the World Wide Web, in the 1990s, the growth of the Internet has been explosive. This growth has meant that the Internet is now ubiquitous and access to this network and the accompanying services that it provides, is available inexpensively to almost everyone. The Internet has evolved from being an interesting network for academic study and for the military, to being an important tool for business, education and the entertainment industry. Several large multinational companies owe their existence and continued profitability to the Internet; amongst these are eBay, Google and Amazon. Large telecommunication companies have seen the value of the Internet and these companies are completing the migration of the traditional telecommunications services offered by them from the large circuit switched networks to the Internet. New services such as Voice over Internet Protocol (VoIP) have been developed to make this migration possible. The way we connect to the Internet has also changed significantly over the past twenty years. Gone are the days of the 300-baud modem; nowadays, broadband connections with speeds in the tens, if not hundreds, of megabits per second are available. The traditional reliance on the copper telephone line, as a means of access is also being challenged. Wireless technologies and infrastructure are being deployed which hold the promise of connection anywhere and anytime. This broadband and wireless access also means that new applications will evolve that use the Internet as their means of moving data from place to place.
Despite this growth, some things have not changed. The Internet is still based on the Internet Protocol (IP), which is a best-effort service. No guarantees are given for the timely delivery of data packets or that the packets will reach their intended destination. Generally, this is not as serious a problem as it first may seem. For example, in the case of data files being transferred across the Internet, it is not of great significance if a data packet is delayed, or even lost, as the applications that handle the data transfer are built to cope with these situations by either waiting for the data packet or by requesting a retransmission. However, some applications do not perform well with a best-effort service; these services are denoted as ‘real time’ services. Several enhancements have been made to minimise the impact of this last issue on real time services, for example, Quality of Service (QoS). Examples of real time services are VoIP and video, these services, are affected by issues such as packet loss, delay and delay variation; however, they can cope reasonably well as long as the loss and delay are not significant.

A third class of real time services are interactive multi-player on-line computer games, these have become more popular and important over the past decade. The most popular genres of on-line games include the First-person Shooter (FPS) and the Massively Multiplayer On-line Games (MMOG)\(^1\) games. FPS games are of particular interest, not only as there are thousands of players (‘gamers’) and servers active on the Internet \(^8\), but also because the nature of these games makes them intolerant of packet loss and delay. One could say that this genre of games is truly real time, as one’s ‘virtual life’ depends on the network performing in a fashion that does not hinder game play. One may ask, “who cares about on-line gamers?” the answer is simple. On-line gaming is now big business; according to DFC-Intelligence, “Total worldwide sales of online games are expected to increase from $19.3 billion in 2010 to $37.9 billion by 2016”\(^{50}\). Several Internet Service Providers (ISPs) host game servers to attract new customers and to retain their current clients, thus, ‘gamers’ are important to them. With the increased popularity of networked multi-player games, the amount of traffic generated by these games is significant, and thus the modelling of this class of traffic has attracted the attention of researchers over the past twenty or so years. Understanding this traffic is useful for ISPs who, as they find themselves in a competitive environment, must deal with the technical challenges in order to provide the best possible service to their customers.

\(^1\)In the past two decades these were more commonly known as Massively Multiplayer On-line Role Playing Games (MMORPGs)
1.2 On-line Multi-player Games

Numerous on-line multi-player games are now available for Personal Computers (PCs), game consoles, and mobile devices. The most popular genres of on-line games are the MMOG and the FPS games. Other popular genres include the Sports, Racing, Strategy, and Simulation. MMOG are on-line games that simultaneously support very large numbers of players. This genre involves large-scale co-operation or competition between players. Popular versions of this genre are ‘World of Warcraft’ and ‘Farmville’. The objectives of these games are to complete quests in order to build up both the experience and strength of the player’s character. The traffic associated with this genre, typically requires low bandwidths, and generally consists of small periodic update packets. This genre does not have strict real-time constraints on the game play [84, 138].

The FPS genre consists of games where the player’s character is engaged in combat with other characters. The viewpoint of the player in an FPS game is such that the player watches the game from this character’s eyes. Most FPS game play is fast, and so this requires the players to react quickly, thus this genre is particularly sensitive to the network conditions, as delayed packets could restrict the quick reactions required in the game. Early examples of the genre are ‘Wolfenstein 3D’ and ‘Doom’. More modern examples usually have multi-player capabilities and therefore they are very popular as on-line games. Several authors have studied the traffic characteristics of this genre, the results of which are presented in Chapter 2. The work in this thesis is based on this genre of on-line games.

Other genres of interest are the sports and racing games, which simulate a competition of some form where one player competes against other players. Caviglione [26] presents results of the traffic produced by ‘Mario Kart Racing’, in this case the traffic consists of very small packets that are sent at regular intervals. The simulation genre includes social simulation games, which are based on social interaction between several avatars. An example of this genre is ‘Second Life’. The traffic characteristics of this last game have been analysed by Antonello et al. [3].

1.3 Motivation

Floyd and Kohler [57] emphasised the need for good traffic models. Just as there are good traffic models for World Wide Web traffic, there is a need for good traffic
models to be developed for FPS games. There have been some attempts at modelling FPS traffic. In 2009, the IEEE 802.16 Broadband Wireless Access Working Group [129] proposed a model for FPS traffic, which is part of the IEEE 802.16m Evaluation Methodology Document (EMD). This model, as well as other models in the literature, is relatively unsophisticated as it does not account for temporal effects nor does it readily model the effects on the traffic due to differing numbers of players.

The motivation for the work in this dissertation is to develop a simple traffic model that can be used to simulate the characteristics of the traffic flowing between a FPS game server and the clients. This model should be general enough so that representative traffic can be generated for different numbers of participants in the game. The model should also include temporal characteristics so that the effects of mixing this type of traffic with other traffic classes can be properly evaluated. A computationally simple traffic model is also desirable, since this can be used as the basis of either hardware or software based traffic generators (such as the one described in the Broadband Internet Traffic Simulation and Synthesis (BITSS) project [79]); this is particularly important if the generator is implemented in hardware. This work is important as it extends the current knowledge of the characteristics of the traffic produced by FPS games. An FPS traffic model should be useful when evaluating new schemes to manage the FPS game traffic, and in studying the interaction between this type of game traffic and other classes of traffic, in terms of the impact of one class of traffic on the other.

1.4 Scope of Research

This thesis addresses the following important questions:

- Which parameters of FPS traffic are useful in terms of formulating a model of FPS traffic?
- What types of statistical distributions are suitable for modelling FPS traffic parameters?
- Can a simple model capture both the statistical nature of the size of the packets and the temporal characteristics of the traffic?
- If a simple model can be found, can it be extrapolated so that representative traffic for games with large numbers of participants is determined from measurements of games with a small number of participants?
1.5 Thesis Outline

- Can a computationally simple technique be used for the extrapolation?

These questions are addressed throughout this work and the main contributions arising from this work are summarised in Chapter 9.

1.5 Thesis Outline

This thesis is organized as follows. Chapter 2 presents a review of the literature on the modelling of traffic produced by FPS games. Chapter 3 briefly outlines the statistical tools used in this study and it outlines the basic methodology that was employed in the study. Chapter 4 presents the characteristics of the traffic traces for seven popular FPS games that form the basis of this study. In this section, the nature of the traffic is studied. Chapter 5 presents an analysis of fitting standard statistical distributions to the traffic traces obtained from various games with the aim of finding suitable distributions that can be used for modelling the parameters of the traffic. Chapter 6 presents results obtained from modelling the time-series behaviour of the seven games. Chapter 7 presents the results from extrapolating the basic models to games with larger numbers of players. Chapter 8 presents the implementation of the traffic model as well as results from an OMNeT++ simulation. Finally, the major findings and contributions are summarised in Chapter 9 along with a suggested outline for further future work.
Chapter 2

First-person Shooter Games Traffic

2.1 Introduction

The past two decades have seen a great deal of interest in modelling the traffic generated by on-line multi-player games [10, 16, 59, 87, 86, 88, 60, 47, 46, 84, 54, 139, 66, 110, 137, 56]. This interest has been driven by the increase in the number of on-line ‘gamers’, the variety of genres of on-line games that are currently available and the introduction of game consoles. The pursuit of modelling this type of traffic is particularly active amongst researchers who are concerned with both the effects of the network on game playability and user experience [36]; and also with the effects that this game traffic may have on other traffic sharing the network. The traffic from FPS games is of particular interest to researchers as it is similar to other real time application’s traffic as the player’s gaming experience, for this particular genre, is generally intolerant of high latency and packet loss. Just as the behaviour of World Wide Web clients is well understood [45, 44], there is a similar need to understand the nature of FPS traffic in order to plan effectively the provisioning of the network, and to predict the performance not only in order to provide a playing experience of high quality, but also to ensure that other traffic on the network is not negatively impacted by the game traffic. So that the server and network capacity can be planned effectively, it is common to use analytical models, simulations or small test beds. However, for this planning process to be effective, the appropriate traffic simulation models need to be developed [57, 61] for the traffic of interest. Producing a traffic model for a class of traffic is not a simple process, as the model can only approximate the true traffic. Many questions about the nature of this class of traffic need to be answered, such as:
• Is the traffic homogeneous?
• Are its characteristics time varying?
• Are the characteristics dependent on a particular genre, game, map, or player?

In the case of FPS traffic, these models can be developed either by taking measurements from real life scenarios, or by examining the source code of the particular game in question. The first option may not be practical, as the measurements would have to cover a wide variety of scenarios in order to obtain satisfactory models, so techniques that can be used to extrapolate the results of the models must be developed. The second option may not be appropriate, as the game’s source code may not be available, because of commercial reasons, whilst the game is still popular. In addition, as is the case for web traffic, the traffic produced by the FPS game is ultimately driven by the behaviour of the human player, so examination of the source code may not provide enough information regarding the traffic generated [23].

The importance of modelling the traffic from FPS games has been acknowledged by the IEEE 802.16 Broadband Wireless Access Working Group, who have included a traffic model for FPS game traffic in the IEEE 802.16m EMD that specifies the methodology used to evaluate IEEE 802.16m based networks [129]. In addition, Shin et al. [114] has also proposed a generator for FPS traffic; however, the details of the analytical model used to build the generator have not been disclosed.

The existing literature covering FPS games explores several issues ranging from the session-level user’s behaviour and experience, server discovery traffic, player geographic distribution, to packet level modelling and the scalability of models to large numbers of players.

This chapter presents a review of the literature around the modelling of FPS game traffic outlining some of the limitations of the modelling thus far. It shows that the FPS models that exist in the literature are quite simple as they neither capture the temporal characteristics nor do they provide simple techniques to model the variation caused by differing numbers of players. This rest of the chapter is organised as follows: Section 2.2 briefly introduces the historical aspects of traffic engineering. Section 2.3 presents a summary of FPS games. Sections 2.4 to 2.7 deal with the traffic characteristics of the different phases during a game. Section 2.8 reviews the existing literature on how the traffic generated by FPS games varies as the number of players varies. Section 2.9 examines the issue of correlation in FPS game traffic and Section 2.10 presents conclusions.
2.2 Traffic Modelling

The work with telephone communications that was pioneered in the early part of the twentieth century by A. K. Erlang gave birth to ‘Teletraffic Engineering’ [99, 15]. Erlang’s work characterised telephone communications by deriving statistical models for the distribution of telephone call arrivals and their respective duration. From empirical measurements, Erlang determined that the call arrivals could be modelled by a Poisson process and that the durations of the calls were exponentially distributed. Advantages of the Poisson model are that superposition applies, superposition of independent Poisson traffic streams can also be modelled by a Poisson process [61], and that it is memoryless. These properties lead to models that are simple to analyse.

Erlang’s findings were important, as these statistical models could be used to design the telephone network in such a way as to ensure that a certain level of QoS could be met.

Whilst Poisson models have been used to design circuit switched telephone networks, they have disadvantages in terms of other types of traffic. One such disadvantage is that they do not capture the burstiness of traffic as their autocorrelation vanishes and thus they usually underestimate parameters such as the delay time, packet loss probability and the required size of the buffers in a real network [105, 106, 78]. These last issues have led to a re-evaluation of the applicability of Poisson traffic models to modern telecommunications systems, as these now carry a variety of different traffic types, ranging from traditional voice to data and multimedia traffic. The assumption that these different types of traffic can be accurately modelled by simple Poisson processes is no longer valid, so more complex models such as Batch Poisson Processes and Markov Modulated Poisson Processes (MMPP) have been proposed to model different classes of traffic (for example MMPP for video [92]).

Telecommunications networks now carry a significant amount of ‘data traffic’, which is bursty by nature. In their seminal paper, Leland et al. [89] showed that ‘data traffic’, exhibited long range dependent characteristics (self-similarity), this discovery led to a shift in the traffic engineering paradigm away from the Poisson model. Later authors such as Crovella and Bestavros [44], showed that there was evidence of self-similarity present in World Wide Web (WWW) traffic; whilst Paxson and Floyd [105, 106] found that the Poisson process was not adequate to model the TCP arrival processes over Wide Area Networks (WANs).
Several approaches to traffic modelling can be found in the literature that attempt to capture the relevant statistical properties of different classes of traffic. The development of these models is important as they allow one to study the behaviour of the traffic in a network by using simulation. Michiel and Laevens [99] classify the models into two broad classes depending on their intended use:

- **Stochastic models**: these capture the statistical properties of a traffic source (source models) as accurately as possible, e.g. the mean, variance, distribution, autocorrelation. The usefulness of these stochastic models lies in their suitability for simulation use.

- **Bounded Traffic models**: these are based on a limited set of information such as the mean and peak rates. These parameters are usually insufficient to describe the traffic completely and thus are not good for simulation, but they have a role in admission control.

This thesis will only consider stochastic models as these allow one to create models that characterise the traffic source. These source models can then be used to simulate the traffic generated by a node, such as in the case of a personal computer running the game client software, and thus they can be used to explore the behaviour of the traffic emanating from the source in a network, as well as interactions with other traffic classes.

Stochastic traffic models can be classified as either stationary (autocorrelation structure is independent of a time shift) or non-stationary [2]. Traffic models can be further classified as either Short-Range Dependent (SRD) or Long-Range Dependent (LRD) dependent. SRD models are based on Markov processes or Regression models. In these cases, the models’ autocorrelation dies off exponentially and it is significant for small time lags. On the other hand, the LRD traffic models such as Fractional Autoregressive Integrated Moving Average (FARIMA) [62] and the Fractional Brownian Motion [95] models have correlations that die off at a slower rate compared to the exponential. These models will be discussed further in Section 2.9.

Fluid flow models are also used in teletraffic modelling. These models assume that the discrete nature of the traffic can be ignored, and the traffic treated as a ‘fluid’. This is particularly useful when analysis the performance of buffers, for example, using the ‘leaky bucket’ model [99]. In this thesis, we are interested in the nature of the packet size and therefore fluid flow models will not be investigated.
2.3 First-person Shooter Games

This thesis is only concerned with the FPS genre of multi-player on-line games. These games typically consist of an Internet hosted game server and corresponding game clients that support either a single human player or multiple players, such as on a game console, refer to Figure 2.1. Each game client acts as an interface between the human player and the virtual game-world within which the player interacts with other players or other game entities. Multi-player on-line games have a fairness requirement, which implies that the game-state information must be shared between the clients in as close to real-time as practical. In principle, the game clients may be designed to communicate directly with each other in a peer-to-peer fashion [19]. However, in practice, most FPS games are based on a client-server model. The server is responsible for ensuring that the game’s rules are adhered to, thus minimizing opportunities for cheating. The game server is also responsible for propagating the game-state information to the clients by sending update packets (or snapshots) to each client at regular intervals (snapshot interval) independently of the level of game activity (see Figure 2.1). The game clients use these packets to update their own versions of the game’s virtual world. Clients in turn also send update packets, which contain their actions, back to the game server. These update packets are sent at less precisely defined intervals, which are often influenced by the client’s hardware configuration and the amount of player activity [87, 88]. This communication mechanism ensures that every client participating in the game is kept up to date with the actions taken by the players on other clients.

2.3.1 FPS Game Traffic Source Models

The traffic models for FPS games presented in the literature are stochastic models that are generally based on empirical measurements rather than being derived from the examination of the game’s source code. Stochastic methods are appropriate for modelling, since the game’s traffic is driven by the behaviour of the human players, which is, to a large degree, non-deterministic. Trying to understand the nature of the traffic by examining the game’s source code is difficult, or impossible, as it is usually not available (due to commercial reasons) whilst the games are still popular. In addition, examination of the source code may not provide enough information regarding the game traffic generated. Branch et al. [21] claim that the Quake III
Figure 2.1: FPS Game Scenario

Arena source code gives an understanding of what information the server sends, but gives little insight into the nature of the traffic generated by the server, except that the server’s output is driven by its input and consequently by the human player’s behaviour. Since the individual player’s behaviour is unpredictable, the analysis of the server code alone cannot be used to completely predict the traffic generated by the server. Stochastic methods are more appropriate for understanding this traffic, rather than relying on the analysis of the code, as these incorporate the non-deterministic nature of game play. However, one area where the examination of the source code can yield useful information is in determining if there are periodicities in the traffic, for example if the server sends periodic heart beats to the clients. These characteristics will usually be manifested as deterministic traffic rather than random phenomena.

A simple general model for FPS game traffic that can be used in simulations is desired. Simple models are desirable as they are parsimonious [108] and they generally lead to shorter run-times when being used in a simulation. The parameters of the model should be easily derived from simple controlled measurements from actual games and by simple means be extended to simulate a wide variety of games.
and game scenarios. Over the last decade, several authors have developed simple source traffic models for many popular FPS games: some examples are: the Quake series [16, 85, 88], Half-Life [87], Half-Life Counter-Strike [59, 60], Halo [86, 139], and Unreal Tournament [68]. More recently, traffic generators and models for generating FPS traffic have been proposed [114, 129]. However, one major shortcoming of these models is that they are entirely empirical. They are based on inferring probability models for the games by identifying the most appropriate statistical distribution that fits the packet traces according to some ‘goodness of fit’ criteria. There have been few attempts at trying to understand why the traffic has particular characteristics both in terms of the statistical distribution of the traffic and the correlations and periodicities present in the traffic.

Another important question to consider is how the traffic for a particular game scales as the number of participants in the game is varied. This is a major shortcoming of many of the models presented in the literature thus far, and many models ignore the effect of player numbers on the traffic (e.g. TGm [129]) yet it is a major factor in predicting how the traffic varies when the number of participants changes. Does the traffic vary in a predictable way as the number of players is changed? The work done by Lang et al. [87, 88], Zander and Armitage [139], Färber [59] suggests that it may. If so, can this characteristic be used to construct a model that uses the number of players as a parameter for determining the characteristics of the traffic? In addition, is it possible to extrapolate the model to games involving large numbers of players from models that are determined from measurements in controlled experiments involving a small number of players? This last issue will be explored further in Section 2.8. Given this evidence, a general model that can incorporate the above qualities is desirable.

2.3.2 Traffic Compression

The development of FPS games, in terms of networking, has been driven by two related issues: bandwidth and delay or latency. As broadband networks are deployed, the bandwidth issue is becoming less significant; however, many existing games were designed assuming that they would be played over consumer access technologies with limited bandwidth, thus this issue is still important. Latency on the other hand, is still difficult to deal with from just the network design point of view. Generally, games will employ some form of latency compensation techniques to deal with this issue [8].
Compression is one possible technique that can be used to mitigate the effects of the delay introduced by the network [8] and the bandwidth limitations. The compression techniques used, range from techniques that minimise the amount of actual bits required to transmit a given amount of data (Huffman coding), that rely on the server only sending information about the changes from the last frame the client confirmed as received (Delta Compression) [80], to techniques where the server only sends information about regions that are currently visible by the client [8, 18]. These compression techniques ensure that the server customizes the client update packets for the intended client. Although this minimizes the size of server to client packets, it also reduces the potential utility of multi-casting the server update packets to every client [18]. As the information is specific for each client, the size of the update packet may be different for each intended client. The in-game activity conveyed in one update packet includes a component proportional to the number of other players and game entities that are visible to a particular client at a given point in time [21, 118, 80]. The actual visibility of the other players and game entities, and the amount of activity, is dependent on the number of players and the layout ('map') of the virtual world. For example, layouts with many walls and corridors will result in less visibility between players (and less information per update packet on average) than maps with large rooms or wide-open areas. Similarly, games with maps that enable many players to congregate in common areas will experience many more player-player interactions (per unit time) than games with only a few players that are scattered around the virtual game world [8, 21]. Thus, one would expect that the traffic generated by the game would vary with the number of players.

More modern game engines, such as the Quake III Arena (Quake3) and Unreal Engine [126], use a ‘Generalized client-server model’, where the server and client execute the same code on similar data in order to minimize the traffic [83]. In this case, the server still maintains the ‘over-all’ game state; however, the clients maintain an accurate subset of the game state, which they use to predict the flow of the game (‘replication’). The server periodically sends updates so that the clients can correct their predictions. This technique minimises the amount of data that flows from the server to the client. The server still needs to inform the clients periodically of the variables that have changed on the server side, so it is reasonable to assume that most of the time the server would send small update packets to the clients, which contain information that is relevant to a particular client. The server would still need to send larger sized packets, albeit less frequently, in situations where the complexity of replication is greater than the benefit obtained from its use. Situations
where this may occur is when a random event occurs, as Sweeney [126] says “Only in the rare case, such as a player getting hit by a rocket, or bumping into an enemy, will the client’s location need to be corrected”. Even with the modern game engines, the traffic depends on the nature of the game play, which is non-deterministic, thus stochastic methods are still appropriate.

The traffic compression that is employed by the game engines increases the non-deterministic nature of the size of the update packets, which in turn increases the desirability of using stochastic methods in modelling this type of traffic. Additionally, the decisions of the human players cannot be predicted with certainty, thus these would also contribute to the non-deterministic nature of the game-play and ultimately on the size of the update packets. Given these issues, it is unlikely that examination of the game’s source code alone would be of great value.

The next sections present an examination of past efforts for modelling FPS games and their associated traffic.

### 2.4 Phases of Game-play and Game Traffic

A typical FPS game consists of various phases of interaction between the client and server with different network traffic characteristics [10, 18]. These phases can be described as follows:

- **Start-up Phase**: The client performs server discovery in order to choose a suitable game server, once complete the 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.). This phase is characterised by large amounts of data flow from the server to the client. The client may have to wait until all other clients are ready to begin the game before proceeding. This phase is not time critical as the players are not engaged in the interactive stage of the game play hence the network can exhibit large latency, jitter and packet loss without significantly affecting the player’s game experience.

- **Game-play Phase**: The client is connected to the server and the game is in progress (the players are interacting with each other). This phase is characterised by an exchange between server and client of small update packets at small regular intervals. The inter-arrival times of these packets are typically short so that realism can be maintained. This phase is sensitive to latency, jitter, and
packet loss with latency (lag) being the most critical parameter that affects the player’s game experience [69, 4, 60].

- Level-change Phase: The client is connected to the server, but the game-play activity has been suspended as the server changes maps or restarts the map. Again, this phase is also characterised by large amounts of data flow from the server to the client, but similar to the start-up phase is not time critical.

2.5 Session Level

Modelling of the session level aims to answer questions such as: “how often do new players join a game?”; “how long do they play for?”; “how many players are playing on a particular server and where do they come from?” These questions need to be addressed if a thorough understanding of the FPS game traffic is required, and so an overview of the issues surrounding the session level is included below. Modelling the session level is important when one is interested in determining the resources required for hosting on-line games, or if a complete model of player behaviour is required [52]. These issues have been tackled by several authors, Henderson and Bhatti [71] studied the session level behaviour of Quake, Quake III Arena and Half-Life players; Chang and Feng [34], Feng et al. [54], Chambers et al. [33, 32] analysed the traces obtained from a Half-Life Counter-Strike server in terms of the session membership. Similar work was conducted by Sinha et al. [115] for the FPS game Half-Life Counter-Strike over Broadband Fixed Wireless (BFW) services and by Armitage [4] for the location of Quake III Arena players. The findings of these authors are summarised below.

The parameters of interest in characterising the session level include:

- Session membership: measures the number of players participating in a game.
- Session inter-arrival time: measures the time between new players joining a game.
- Session duration time: measures how long a player remains in a game.
- Geographical distribution of the players.

These parameters are discussed in more detail in the following sections.

2.5.1 Session membership

The session membership characterises the number of players that are participating in a game at any given time. The session membership for Quake and Quake III
2.5 Session Level

Arena [71], Half-Life [32, 31], Half-Life Counter-Strike [115] and Wolfenstein Enemy Territory Pro [140, 5, 6] show significant seasonal effects, as the number of participants show both time of day variations, generally with peaks in the afternoon, and day of the week variations with peaks around the weekend. Henderson and Bhatti [71] show that a seasonal ARIMA$(2, 1, 1)(0, 1, 1)_{48}$ with a drift, models the session membership well. Sinha et al. [115] also observed a seasonal variation in the session membership for Half-Life Counter-Strike; they found that the daily peaks occurred at midnight and that the weekly peak occurred on the weekend. Furthermore, they suggest that a time-series model (ARIMA) would be a suitable model; however, they do not fit one to their data. Armitage [5] also observes a seasonal variation in the session membership, and he concludes that the variations depend on the local time as most players begin joining in the afternoon and continue joining into the evening, again no analytical model is proposed.

2.5.2 Session Inter-arrivals

The session inter-arrival time characterises the time between new players joining a game. Henderson and Bhatti [71] show that the session inter-arrival times for Quake and Quake III Arena are significantly auto-correlated, implying that the arrival of players leads to other players wanting to join the game. By using a ‘Hill estimator’ technique, they show that these inter-arrival times have a heavy-tailed distribution and are not independent, thus the underlying process cannot be a Poisson process. Similarly Sinha et al. [115] also show that for Half-Life Counter-Strike the inter-arrivals distribution is heavy-tailed. The session inter-arrivals show seasonal effects and are clustered between Friday evenings and Sunday nights, indicating that many gamers start playing on weekends rather than on weekdays. There is also a period between late night and early morning in a day, when there are very few to no gamers are playing [115].

2.5.3 Session Duration

The session duration characterises the time that players remain in a game. This parameter is useful when developing source models of FPS games, as it models the client’s ‘on-time’. Several authors [34, 31, 32, 55] claim that the distribution of the session duration for Half-Life Counter-Strike is heavily dependent on the quality of the game server, whilst Henderson and Bhatti [71] find that for Half-Life, Quake III Arena and Quake, there does not seem to be a correlation between
the session duration and the session membership. They claim that the median session duration is relatively constant irrespective of the number of players; hence, the absolute number of players in a session is not necessarily a determining factor of when a player decides to leave a session. They claim that the distribution of the session duration is exponential. The results from Sinha et al. [115]'s work do not support this claim for Half-Life Counter-Strike. Their results show that the session duration process is very bursty and variable with durations ranging from 90 seconds to 13 hours with mean of 775 seconds. In this case, they claim that the session durations are heavy-tailed. Possible reasons for the difference between Sinha et al. [115] and Henderson and Bhatti [71] are attributed to the different methods used to collect the traffic traces (e.g. granularity), the difference in the number of servers used as collection points, the population size and the claim that the user’s behaviour may have changed over the time that the two studies were conducted (more than 2 years). In contrast, Feng et al. [54], Chambers et al. [31] find that for Half-Life Counter-Strike a significant proportion of the session durations are very short and that the number of players that play for longer periods of time drops sharply as time goes by. Their claim is that the session duration is decidedly not heavy-tailed, but can be modelled by a Weibull distribution with a shape parameter $\beta = 0.5$, a scale parameter $\eta = 20$, and location parameter $\gamma = 0$ [54] as the session times model the ‘player lifetimes’ or ‘failures’. However, it should be noted that no results of a goodness-of-fit test for the fitted distribution were presented in this work. Possible reasons for the difference between these results and those of other authors are attributed to the claim that Half-Life Counter-Strike servers are “notoriously heterogeneous” [54], i.e. short durations may correspond to players browsing the server’s features, this behaviour is not typical of other games. Another possible reason was attributed to the game server configuration, where a ‘plug-in’ was used to remove players who continuously killed their team-mates; this feature may have led to players being quickly removed from the server’s game session. In a more recent paper, Chambers et al. [32] claim that the session times for Half-Life are ‘heavy-tailed’ and that the session duration can be modelled by Weibull distribution with a shape parameter $\beta = 0.456$, a scale parameter $\eta = 11.7$, this is supported by previous work in Feng et al. [54] and Chambers et al. [31].

Hariri et al. [67] propose a different approach based on a hierarchical hidden Markov model to the modelling of the session duration of FPS games. In this approach, Hariri et al. [67] use a three level hierarchical Markov model, one to model the game state transitions, followed by another Markov model to model the
actions of a single player, and a third Markov model to model the generation of the message updates between the client and server. This Hierarchical model was trained by using available data from results published for various FPS games. Their model is used to predict the probability of a player leaving a game as a function of the time that a player stays in the game, as well as the probability of a player winning the game. There is not much analysis or discussion regarding their results, except to say that their “results seem to be compatible with the measurement results for session duration” [67].

2.5.4 Geographical Distribution of Servers and Players

As competitive online FPS gameplay typically requires latencies (lag) below 150–200 ms [4, 60], FPS game players prefer game servers with low latencies [72]. The latency between the player’s client and the server is typically determined by the geographical distance between the players and the server, as well as the various serialisation and queuing delays in the network. Based on this last assertion, one would expect that on average players would choose servers that are close (geographically) to them. Knowledge of the geographical distribution could lead to better strategies for server discovery hence reducing the traffic due to this process. The geographical distribution of players was initially studied by Feng and Feng [55] who determined the geographic locations the players connecting to a server hosting Half-Life Counter-Strike, Battlefield 1942, and Unreal Tournament 2003 games. Their motivation for this work was to explore issues such as: the current location of game servers, the importance of geographic proximity to game servers, and whether the geographic distribution of game servers are correlated with the geographic distribution of game players. Although the results of this study may no longer be accurate, as the distribution of the servers and the players may have changed significantly since the publication of their original work, they presented some interesting conclusions. Feng and Feng [55] found that a large percentage (45%) of players would connect to servers that were geographically remote from them. Later work by Zander et al. [140] also showed that 20 – 30% of the players connecting to a Wolfenstein Enemy Territory Pro server were not local players. Of the possible explanations, that Feng and Feng [55] outline for this observation, two are of interest. The first is the claim that “the number of players on a server determines desirability over delay”. If this statement is, in fact, true, then lag alone does not determine the session membership. This observation is supported by Henderson and Bhatti [71] who showed that
there was significant autocorrelation in session inter-arrival times, implying that the arrival of some players will lead to other players wanting to join the game. The second observation of interest is that there was a significant time of day factor in the geographic distribution of players. Feng and Feng [55] found that the average distance from the server had periodic behaviour and that the average distance was highest in the early morning and reached its minimum during late afternoon and evening hours, refer to Figure 2.2. Chambers et al. [30], use the results of this work to define a redirection service for game servers that connects players to servers that are geographically close. It is appropriate that this behaviour be modelled so that the seasonal effects and the random effects of the where the traffic originates and ends can be separated.

Armitage [5], on the other hand, uses an approach based on server discovery probes to determine the geographic distribution of the players. This method has the advantage that the probe traffic occurs 24-hours per day irrespective of the popularity of the server and is independent of the game-play traffic [140]. Armitage [5] also found a 24-hour cyclical variation in the session membership that tracks the time of day.

In summary, the geographical location of players follows a 24-hour cycle with players choosing servers that are not necessarily geographically close to them, other criteria, such as the popularity of the server, may be important factors in the player’s choice of server.

Although modelling the session level is important and well understood, the discussion of this modelling has been presented for completeness sake; this research will be concerned with the game-play phase, as this is where the network conditions affect the player’s gaming experience.

2.6 Modelling the Start-up Phase

The start-up phase usually consists of the game client performing server discovery so that a suitable FPS game server can be located, connecting to this server and downloading the appropriate game information and files from the server in order to be able to play the game [70]. Most of the work done in characterising this phase has
2.6 Modelling the Start-up Phase

2.6.1 Server Discovery Process

Server discovery is the process by which game clients locate up-to-date information about the game servers that are available at any given time for a particular FPS game [6]. Using this process, the player can select the most appropriate server on which to play the game. The server discovery process is similar for many types of online FPS games where there is a centralised global list of known active game servers stored on a master server. The client fetches this global list so that the human player can choose the most appropriate game servers on which to play based on criteria such as the number of current players and latency between the player’s client and the server. This process was investigated by Armitage [6] who claims that due to the large number of FPS game servers that are active on the Internet at any given time, the challenge for the designers of FPS game is to make the server discovery process use minimal network resources whilst still providing accurate and appropriate information back to the client.

2.6.2 Server Discover Traffic

Server discovery consists of the client requesting and obtaining, a list of all IP addresses and port numbers representing game servers that are registered as being currently active with the master server. On obtaining this list, the client sequentially probes each listed game server for information about the nature of the game. These probes typically consist of an exchange of UDP packets, which are also used to provide an estimate of the Round-trip Time (RTT) between client and server at the time of the probe [6]. The human player can then use the information obtained from the probes to select the ‘best’ game server on which to play. This process is illustrated in Figure 2.3.

The client normally sends a large number of probe packets in order to find and connect to a single game server, and so a particular server receives a large amount of probe traffic regardless of the number of players actually playing on it or its
popularity. Armitage [5] refers to the server discovery probe traffic as ‘background noise’ as it results in a continual influx of probe packets to individual game servers, 24-hours a day and 7 days a week. The probe traffic for Wolfenstein Enemy Territory [5, 6] and Counter-Strike-Source [7] fluctuates on a 24-hour cycle that follows the waking and sleeping cycle of population centres around the world, and continues as long as a game server is registered with the master server [6]. This observation is consistent with the work by Zander et al. [140] who found that the probe traffic was not necessarily correlated with the actual game-play traffic patterns.

Armitage [5] identifies trends in Wolfenstein Enemy Territory server discovery traffic. He claims that the total number of Wolfenstein Enemy Territory clients being turned on and off each day was slowly declining during the first half of 2006. However, he does not try to model this trend. Armitage also finds that server discovery probe traffic is seasonal as it varies over a 24-hour cycle, and that it may have a seasonal trend since the peak probing hour shifts between winter and summer as players change their daily patterns of play [5], however, no seasonal model has been proposed.

Armitage and Branch have shown that the server discovery probe traffic for Wolfenstein Enemy Territory Pro [5] and Half-Life 2 [7] can be modelled by a Poisson process, (probe arrivals are uncorrelated and inter-probe intervals are exponentially distributed) for both the busiest and least-busy hours of the daily cycle. The rate parameter, which is derived from the median of the inter-probe interval from the empirical data during the hours of interest, is shown in equation 2.1:

\[ \lambda = \frac{\ln(2)}{\text{MedianInterval}} \] (2.1)

The median interval parameter for any hour of the day is estimated using a modified Laplace curve as shown in Figure 2.4. This Poisson model can be used to synthesise probe traffic patterns that can be used in modelling the traffic experienced by game servers and other devices attached to them.
2.7 Modelling the Game-play Phase

The game-play phase is sensitive to latency (lag), jitter, and packet loss with latency being the most critical parameter that affects the player’s game experience [4]. Much of the literature is concerned with modelling the traffic during this phase of the game and several authors have presented models for this traffic. As mentioned in Sections 2.3.1 and 2.3.2 stochastic methods are appropriate and they have been used widely to understand and model the traffic characteristics of this phase in FPS games, see Ratti et al. [113] for a summary. The traffic parameters of interest during this phase are:

- Client-to-Server (C2S) and Server-to-Client (S2C) Inter-arrival times (IATs). These parameters model the time between successive packets flowing in the upstream direction (Client-to-Server) or the downstream direction (Server-to-Client).
- Client-to-Server and Server-to-Client payload size. These parameters are measures of the size of the User Datagram Protocol (UDP) payload in either direction (refer to Figure 2.5). This parameter will be referred to as the ‘payload size’ in the rest of this thesis.

Ratti et al. [113] provides details of the values of these parameters for different FPS games. Some of the more important ones are summarised in subsequent sections of this work. Often these parameters are measured in different network environments [141], so discrepancies in their values may be found in the literature. Some authors also quote the bandwidth and or data rates in packets per second (pps) [86, 88, 54, 87]; however, as these parameters can be derived from the IAT and payload sizes they are not considered in this work.
2.7.1 Probability Density Function Based Traffic Modelling

Early attempts at describing the traffic produced by FPS games concentrated on the bursty nature of the traffic produced in the different phases of the game-play. Bangun and Beadle [10] found that the traffic between the game server and the clients is bursty and highly asymmetrical, with the rate of the server-to-client traffic being two orders of magnitude larger than the client-to-server rate. Although Bangun and Beadle describe the traffic characteristic of their proposed ‘on-demand multi-player scenario’ they do not present a traffic model that describes the statistical distribution of the traffic parameters, but they only quantify descriptive statistics such as mean, standard deviation and burstiness of the traffic. Borella [16] addresses this last issue by examining several empirical packet traces, collected over a Local Area Network (LAN), for the Quake game and fitting appropriate standard distributions to the packet size and inter-arrival time observations. His methodology leads to relatively simple traffic models, where the packet size and inter-arrival times are drawn from standard distributions. The importance of Borella’s work is that he outlines both a methodology and analysis techniques that are subsequently used by other authors to study the traffic characteristics of other FPS games. Borella’s methodology is summarised below.

From the empirical traffic traces collected from the FPS game:

- A standard distribution is chosen based on the examination of the traffic’s empirical Probability Density Function (PDF) or Cumulative Distribution Function (CDF). Where the empirical distribution has ‘spikes’, these are modelled by degenerate or deterministic distributions. The chosen distribution’s parameters are estimated by using a Maximum Likelihood Estimation (MLE) method.
2.7 Modelling the Game-play Phase

- The fit of the distribution is checked by either comparing the analytical and empirical PDFs and or CDFs, or by plotting a Quantile-Quantile Plot (Q-Q). Where there is a significant deviation between the empirical and analytical distributions, Borella suggests that the data set should be spilt into parts and each part be modelled by separate distributions.

- A goodness-of-fit metric should be used to determine how well the model agrees with the empirical data. Borella suggests the use of the $\lambda^2$ discrepancy measure ($\lambda^2$) [107] as he claims that traditional goodness-of-fit tests, such as Chi-square ($\chi^2$) and Kolmogorov-Smirnov (K-S), often do not work well with large data sets that are autocorrelated.

As Borella’s methodology has been used by several other researches to model FPS game traffic, it is reasonable to use it as the basis of collecting empirical data on which the traffic modelling can be based. This approach, modified where appropriate, will be used in the rest of this work. The next sections present a description of the traffic parameters of interest in modelling the game-play phase.

A point of confusion may arise with the terminology used by several authors when describing the statistics of the update packet size. The payload size of the update packet is discrete thus it should be modelled by a Probability Mass Function (PMF); however, several authors use the PDF, which strictly speaking, refers to a continuous rather than to a discrete distribution. This is not a major problem, as in general the distribution would be used as part of a simulator in which case it is understood that the random variable describing the packet size would need to be converted to an integer value.

2.7.2 Assumptions in First Person Shooter FPS Game Traffic Modelling

Much of the FPS game traffic modelling follows Borella’s methodology, and builds on it by producing client and server traffic models for different FPS games. In the interests of simplicity, these models are based on certain underlying assumptions that are outlined by Färber [59] in his traffic model for Half-Life Counter-Strike:

1. Each client’s behaviour is independent of the other clients in the game.
2. Server traffic per client is independent of the number of clients.
3. Traffic produced by the clients is independent of the corresponding server traffic.
4. The traffic during game interruptions is not time critical.
Whilst these assumptions are inherent in both previous and subsequent work, stating these assumptions explicitly allows one to question their implications for the traffic models. The first assumption is important and fundamental as it allows for the implementation of separate client traffic generators that draw their statistics from identical distributions; however, this assumption also implies that there is no cross-correlation in the traffic generated by the clients. This assumption may not be valid as one would expect that players’ interactions during the game-play phase would influence the traffic produced [11]. This assumption also implies that the traffic streams from each of the clients to the server are also independent. The second assumption suggests that the server traffic model can be determined from any game irrespective of the number of players participating in the game. This assumption will also ultimately lead one to stipulate that the traffic produced by games with a large number of players can be inferred from that produced by games with a small number of players. The third assumption justifies the splitting the traffic model into separate client and a server traffic generators. Whilst the fourth assumption suggests that only the game-play phase is the most interesting one to model, as this is where the real time constraints are imposed on game play.

Other important assumptions that are made in the modelling of FPS game traffic are:

1. Fairness: The protocols used in the game are designed so that they are fair for all players [20].
2. Stationarity: The traffic characteristics are stationary. This implies that the statistics of the game do not change with time.

These assumptions may seem reasonable in order to produce simple models, but they must be kept in mind when producing FPS game traffic models. The second assumption that the traffic characteristics are stationary is particularly important, yet many authors make this assumption without ensuring that their data is in fact stationary, e.g. [87, 88]. A number of the above assumptions will be used for deriving models in latter sections of this work, however where appropriate their validity will be tested.

In the next section, we present a review of the FPS traffic characteristics during the game-play phase. Descriptions of the various statistical distributions used by various authors to model the FPS game traffic can be found in Section 3.3.
2.7 Modelling the Game-play Phase

2.7.3 Client-to-Server Inter-arrival time

The term client-to-server inter-arrival time will be used in this work to denote the time interval between successive update packets flowing from the client to the server (refer to Figure 2.5). The fairness assumption dictates that each client must report its ‘state’ frequently so that the server can ensure that the global state information regarding the location and actions of each player is current. This requires that clients send their own updates at some specified minimum rate. Consequently, these updates typically occur at a fixed periodic rate, generally every 10 ms to 50 ms. For some FPS games, this time interval can be configured at either the client or server. Thus, one would expect that the distribution of the client-to-server IATs, under ideal circumstances, would be described by a degenerate distribution (impulse), this is confirmed by Färber [59] for Half-Life Counter-Strike. The results from the literature suggest that this parameter is deterministic and many researches use distributions with small variances such as the Uniform distribution for the game Halo [86], or a Gaussian distribution for Halo II [139] to approximate a degenerate distribution, or unspecified deterministic distributions [48]. Other authors have modelled this parameter by a Uniform distribution for Open Arena and Team Fortress [51]. Some empirical measurement studies have reported that the client-to-server inter-arrival time may depend on factors such as the client’s processor speed, the graphics card, hardware configuration, and player activity [88]. These issues introduce some degree of randomness and so the observed distribution of the client-to-server inter-arrival times in practice is not a true degenerate distribution. Other authors have modelled the client-to-server IAT parameter by using a mixture of distributions. For example, Borella [16] chooses the Extreme Value and Exponential distributions to model Quake, Lakkakorpi et al. [85] uses a mixture of Extreme Value and Gaussian distributions for Quake II, whilst Lang et al. [88] and [102] use discrete and exponential distributions for Quake III Arena. As this parameter is not always measured under controlled conditions on a dedicated test-bed, other effects, which may have been due to interference from other traffic on the network, may have introduced a degree of randomness in this parameter. Another issue that may have contributed to the randomness of this parameter is that of the time-stamping accuracy of the measuring devices, it is difficult to determine exactly how accurate the time-stamping was in the above studies and how this has affected the randomness of the measured results. To avoid this last problem, the time-stamping accuracy must be determined when measuring the inter-arrival times.
## Table 2.1: Client-to-Server Inter-arrival time

<table>
<thead>
<tr>
<th>Game</th>
<th>Client-to-Server Inter-arrival time (ms)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Quake</em> Q1 Trace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client Type 1</td>
<td>13 (lower 50%), exponential(19.61)</td>
<td>Inter-packet interval depends on the client’s hardware. [16]</td>
</tr>
<tr>
<td>Client Type 2</td>
<td>(upper 50%)</td>
<td></td>
</tr>
<tr>
<td>Client Type 3</td>
<td>extreme(a=20.63, b=4.39)</td>
<td></td>
</tr>
<tr>
<td>Client Type 3</td>
<td>extreme(a=23.06, b=5.23)</td>
<td></td>
</tr>
<tr>
<td>Client Type 4</td>
<td>14 (lower 50%), exponential(18.38)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(upper 50%)</td>
<td></td>
</tr>
</tbody>
</table>

| *Quake* Q2 Trace |                                         |                                                                          |
| Client Type 1   | extreme(a=36.38, b=8.01)                | Inter-packet interval depends on the client’s hardware [16]              |
| Client Type 2   | extreme(a=15.89, b=2.98)                |                                                                          |
| Client Type 3   | extreme(a=30.26, b=6.30)                |                                                                          |
| Client Type 4   | 14 (lower 50%), exponential(18.38)      |                                                                          |
|               | (upper 50%)                             |                                                                          |

| *Quake* II      | Lower 4.5%, x < 18: Extreme(a=6.57, b=0.517) | Determined by the cl_maxfps parameter [85]                                |
|                | Upper 95.5%, x>18: Extreme (a=37.9, b=7.22) |                                                                          |

| *Half-Life* Counter-Strike | 40 | [59] | 50 | [60] |

| *Half-Life* | Open GL: Software Clients: | 33 (lower 50%) 50 (upper 50%) 41 | Depends on the type of graphics card used by the client host [87] |

| *Quake* III Arena | Intel 845G card: | 60% 10.75 40% (10.75 + exponential(4.29)) | Depends on the type of graphics card used by the client host [88] |
|                  | NvidiaGF2 card:  | 16% 10.75 84% (10.75 + exponential(5.85)) |                                                                          |

| *Halo*          | 40 ms + uniform(0,1) μs | [86] |
|                 | 201 ms (periodic fixed 72 byte packet) | |

| *Halo* II       | normal(40,1)           | [139] |

| *Quake* 3       | 11                       | [102] |

| *Team Fortress* | uniform (31,42)          | [51] |

| *Open Arena*    | uniform (69,103)         | [51] |

It is noteworthy that some authors choose the Gaussian distribution or other distributions whose support is the entire set of real numbers, for modelling the client-to-server inter-arrival times. This should be approached with care, since there is a non-zero probability of producing negative values of inter-arrival times.

The results of the client-to-server inter-arrival time for various FPS games are presented in Table 2.1. A similar approach to that taken by other authors will be taken for modelling this parameter in later sections of this work.
2.7 Modelling the Game-play Phase

2.7.4 Client-to-Server (C2S) Payload Size

The term client-to-server payload size will be used to denote the size of the payload of the update packets flowing from the client to the server (refer to Figure 2.5). The players involved in a FPS game interact with their surroundings and the other players by performing certain actions. The number of actions is limited and typically, they involve the player moving, fighting or collecting objects. Some actions, for example movement, are more frequent than others, such as fighting. Branch and Armitage [18, 20] argued that as players can only perform a limited number of actions in a short period of time, and since certain actions would occur more frequently, the client-to-server payload size PDF should consist of a number of impulses that span a small range, which represent the payload size of the set of actions. This statement is supported by the empirical model of Quake II put forward by Lakkakorpi et al. [85] and of Degrande et al. [48] who claim that the client-to-server payload size can be modelled by discrete distributions. As mentioned previously, the term PDF rather than PMF will be used even though this distribution is discrete. Typically, the sizes of the client-to-server payload vary over a small range and are modelled by either Gaussian distributions with small variances Half-Life [87], Quake III Arena [88, 102], Open Arena [51], or extreme distributions Half-Life Counter-Strike [59]. However, some of the empirical evidence shows that the client-to-server payload is of a fixed size for some games, e.g. Quake I [16]. Färber [59] found that the client-to-server payload size is independent of the number of clients in the game; however, in the case of the Halo and Halo II games the payload size varies with the number of players active on a particular client [139].

The results of the client-to-server payload sizes for various FPS games are presented in Table 2.2. In summary, this parameter, which is an essential part of the traffic model, varies in size and must be modelled by a distribution that is appropriate for a particular FPS game. This issue will be examined further in Chapters 4 and 5.

2.7.5 Server-to-Client IAT

The term server-to-client inter-arrival time will be used to denote the time interval between successive update packets flowing from the server to the client (refer to Figure 2.5). To ensure that each client is treated fairly by the server, the server will generally send update packets to each client in a back-to-back burst ([87] [88]), thus, each client receives regular updates irrespective of the level of the players’ activity in the game. Typically, the server sends these updates every 30 to 60 ms;
### Table 2.2: Client-to-Server Payload Size

<table>
<thead>
<tr>
<th>Game</th>
<th>Client-to-Server Payload Size (Bytes)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake</td>
<td>24</td>
<td>Fixed Size Packets [16]</td>
</tr>
<tr>
<td>Quake II</td>
<td>Discrete Distribution: 10.6%: 36, 26.4%: 42, 6.26%: 44, 13.9%: 45, 4.95%: 46, 16.3%: 48, 21.5%: 51</td>
<td>Payload and UDP header [85]</td>
</tr>
<tr>
<td>Half-Life</td>
<td>Extreme (a=80,b=5.7)</td>
<td>Range: 20 to 100 bytes [59, 60]</td>
</tr>
<tr>
<td>Counter-Strike</td>
<td>Extreme (41.6)</td>
<td></td>
</tr>
<tr>
<td>Half-Life</td>
<td>normal (72.29, 6.97)</td>
<td>Size of IP datagrams [87]</td>
</tr>
<tr>
<td>Quake III</td>
<td>normal(64.15, 3.20)</td>
<td>Size of IP datagrams [88]</td>
</tr>
<tr>
<td>Arena</td>
<td>80 + 30 * n every 40 ms</td>
<td>Size of IP datagrams [86]</td>
</tr>
<tr>
<td></td>
<td>72 every 201 ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>n=number of players on client Xbox</td>
<td></td>
</tr>
<tr>
<td>Halo II</td>
<td>1 player: extreme(a=71.2, b=5.7)</td>
<td>Extreme distribution parameters depend on number of players on client Xbox [139]</td>
</tr>
<tr>
<td></td>
<td>2 players: extreme(a=86.9, b=5.1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 players: extreme(a=111.5, b=7.7)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 players: extreme(a=127.7, b=8.2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(rounded to nearest 8 byte packet size)</td>
<td></td>
</tr>
<tr>
<td>Open Arena</td>
<td>Gauss (42.2;4.6)</td>
<td>[51]</td>
</tr>
<tr>
<td>Team Fortress</td>
<td>Gauss (76.52;13.9)</td>
<td>[51]</td>
</tr>
</tbody>
</table>
in certain FPS games this parameter is configurable. The precise interval depends on the game being played and in some cases on the snapshot rate requested by the client [8, p. 164]. Again, as these updates are sent at a regular fixed interval, it is reasonable to expect that this traffic parameter will show little variation and that it is adequately modelled by a degenerate distribution [48], this is the case for Half-Life Counter-Strike (Färber [59]), Unreal Tournament 2003 [14], Quake III Arena ([88] [102]) and Halo ([86]). Any variation in this parameter is usually caused by variations in either the server’s level of activity, or by the accuracy of the server’s clock [8, p. 163] or due to delays in the network. As was the case with the client-to-server inter-arrival time, authors generally use distributions with small variances to model this parameter, for example Halo II [139], and Call of Duty 2 version 1.0 [29]. Others use skewed distributions with larger variances, Quake [16], Quake II [85], IEEE 802.16m Evaluation Methodology Document [129]. Drajic et al. [51] use a uniform distribution to model this parameter for Team Fortress and Open Arena. Lang et al. [87] find that for Half-Life the distribution of the server-to-client inter-arrival time for a particular map is bimodal (50ms and 70ms). Armitage et al. [8, p. 155] explains that this is due to the client requesting updates less frequently than the maximum rate configured for the server and thus the server combines the update times to match the requested rate.

The results of the S2C inter-packet intervals for various FPS games are presented in Table 2.3. The literature shows that this parameter must be modelled by a distribution that is appropriate for a particular FPS game. This issue will be examined further in Chapters 4 and 5.

2.7.6 Server-to-Client Payload Size.

The term server-to-client payload size will be used in this work to denote the size of update packets flowing from the server to the client (refer to Figure 2.5). While most of the client-to-server packets are either of fixed length or show very little variation in size and are independent of the number of clients and or players involved in the game [59], the server-to-client packets exhibit substantial variations in the size of their payload. Table 2.4 summarises the results obtained for various games in the literature. It can be seen that there is considerable variation in the payload size of the server update packet, and for some games we see that the variation in size is dependent on the number of players participating in the game, see also Feng et al. [54]. This large variation makes this traffic ‘interesting’.
Table 2.3: Server-to-Client Inter-arrival time

<table>
<thead>
<tr>
<th>Game</th>
<th>Server-to-Client Inter-arrival time (ms)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake</td>
<td>extreme (44.2, 9.52)</td>
<td>3 clients</td>
</tr>
<tr>
<td></td>
<td>extreme (22.84, 5.29)</td>
<td>4 clients [16]</td>
</tr>
<tr>
<td>Quake II</td>
<td>Lower 4.8%, x &lt; 60: Extreme(a=58.2, b=7.47)</td>
<td>Default 100ms [85]</td>
</tr>
<tr>
<td></td>
<td>Upper 95.2%, x ≥ 60: Normal (a=100, b=17.7)</td>
<td></td>
</tr>
<tr>
<td>Half-Life</td>
<td>Extreme (a=55, b=6)</td>
<td>[59, 60]</td>
</tr>
<tr>
<td>Counter-Strike</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Half-Life</td>
<td>except for rats3 map approximately 50% 50ms &amp; 50% 70ms</td>
<td>[87]</td>
</tr>
<tr>
<td>Quake III</td>
<td>50</td>
<td>Lang et al. [88], Park et al. [102]</td>
</tr>
<tr>
<td>Arena</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halo</td>
<td>40</td>
<td>[86]</td>
</tr>
<tr>
<td>Halo II</td>
<td>extreme (39.7, 1.9)</td>
<td>[139]</td>
</tr>
<tr>
<td>Open Arena</td>
<td>uniform (41, 47)</td>
<td>[51]</td>
</tr>
<tr>
<td>Team Fortress</td>
<td>uniform (39, 46)</td>
<td>[51]</td>
</tr>
</tbody>
</table>

The server may also send small periodic packets of particular sizes to the clients, these generally cause ‘spikes’ in the payload size distributions. Borella suggests that these should be modelled separately by degenerate distributions. Typically, the distribution of the server-to-client payload size is skewed and thus they are generally modelled by either Extreme Value distributions for Quake and Quake II [16, 85], Half-Life Counter-Strike [59], Halo II [139] and the FPS game model for IEEE 802.16m Evaluation Methodology Document [129] or a Log-Normal distribution as for Half-Life [87], Quake III Arena [88, 102], whilst Degrande et al. [48] use the Erlang distribution to fit the tail of the empirical distribution. Drajic et al. [51] use a Gaussian distribution to model this parameter for Team Fortress and Open Arena. Other authors such as Joyce [81] Quake World and Unreal Tournament, Bussiere and Zander [25] Wolfenstein Enemy Territory Pro, Pavlicic and Armitage [103] for Quake III Arena, and Shin et al. [114] for Left 4 Dead, all present statistics for the inter-arrival times and packet lengths; however, they do not fit standard distributions to these parameters.

The Extreme Value and Log-Normal distributions are good choices in terms of fitting the observed data because of their ability to capture the skewness in the
Table 2.4: Server-to-Client Payload Size

<table>
<thead>
<tr>
<th>Game</th>
<th>Server-to-Client Packet Payload Size (Bytes)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake</td>
<td>3 clients extreme (89.92, 34.84)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 clients extreme (65.58, 19.12)</td>
<td></td>
</tr>
<tr>
<td>Quake II</td>
<td>Lower 27.6%, x &lt; 55: Extreme (a=46.7, b=4.39)</td>
<td>Packet size contains payload and UDP header. [16]</td>
</tr>
<tr>
<td></td>
<td>Upper 72.4%, x &gt; 55: Extreme (a=79.7, b=11.3)</td>
<td></td>
</tr>
<tr>
<td>Half-Life</td>
<td>Extreme (a=120, b=36)</td>
<td>Depends on number of players</td>
</tr>
<tr>
<td>Counter-Strike</td>
<td>Extreme (a(n), b(n))</td>
<td>(n). Range 20:1000 bytes.</td>
</tr>
<tr>
<td></td>
<td>a(n) = 34.5 + 4.2n</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b(n) = 9 + 3n</td>
<td></td>
</tr>
<tr>
<td>Half-Life</td>
<td>Map:</td>
<td>Size of IP datagrams.</td>
</tr>
<tr>
<td></td>
<td>chillDM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>m = 202.92, s = 0.31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Odyssey</td>
<td></td>
</tr>
<tr>
<td></td>
<td>m = 154.15, s = 0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rats3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>m = 129.63, s = 0.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Xflight</td>
<td></td>
</tr>
<tr>
<td></td>
<td>m = 109.77, s = 0.24</td>
<td></td>
</tr>
<tr>
<td>Quake III</td>
<td>2 players: lognormal (79.34, 0.245)</td>
<td>Size of IP datagrams.</td>
</tr>
<tr>
<td>Arena</td>
<td>plus an exponential (13) for every additional player.</td>
<td>Depends on number of players. [88]</td>
</tr>
<tr>
<td>Halo</td>
<td>100 + 30 * N</td>
<td>Size of IP datagrams.</td>
</tr>
<tr>
<td></td>
<td>N = total number of players</td>
<td>Depends on number of players.</td>
</tr>
<tr>
<td>Halo II</td>
<td>extreme (126.9, 20.4) 3 players</td>
<td>Depends on total number of players. [86]</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>extreme (271, 33) 11 players</td>
<td></td>
</tr>
<tr>
<td>Open Arena</td>
<td>normal (0.172; 0.05) kB</td>
<td></td>
</tr>
<tr>
<td>Team Fortress</td>
<td>normal (0.241; 0.06) kB</td>
<td></td>
</tr>
<tr>
<td>Quake III</td>
<td>lognormal (m, s)</td>
<td></td>
</tr>
<tr>
<td>Arena</td>
<td>m(n) = 13n + 65</td>
<td>n ≥ 2 Range 60 to 20n + 100</td>
</tr>
<tr>
<td></td>
<td>s(n) = 2n + 7</td>
<td></td>
</tr>
</tbody>
</table>
distribution of the random variables. However, to date there have been limited attempts at describing why the underlying process leads to packet sizes that are either Extreme Value or Log-Normal distributed. The question of whether other distributions exist that have advantages in modelling this parameter over those used previously must be considered. This topic will be explored further in Chapter 5.

2.7.7 Summary

In summary, most authors find the server-to-client IATs are deterministic, and that, the server-to-client payload has considerably more variation than the client-to-server payload does.

The packet inter-arrival times of several FPS games have been studied by several authors; refer to sections 2.7.3 and 2.7.5 for a more complete discussion. As most authors have found that the inter-arrival times of the packets in the server-to-client direction are modelled well by degenerate or discrete distributions, since these times show little variation. Usually the update period is configurable so the inter-arrival times are deterministic with some small random component that is due to either the time-stamping accuracy or the various network delays. In the client-to-server direction, the inter-arrival times show some variation, this is typically due to the platform’s hardware. The payload sizes, on the other hand, are usually modelled by positively skewed distributions. Therefore, these parameters are of particular interest when modelling the traffic. Many of the models for the payload size distributions of the update packets found in the literature are based on the Extreme Value or Log-Normal distributions. However, there has been little discussion justifying why the payloads would be distributed in this manner. An important question that remains unanswered at this stage is whether other distributions, whose properties lead to better models, can be found that fit the empirical data as well as those previously used. Modelling the variation in server-to-client and client-to-server payload size will be addressed further in Chapters 4 and 5.

2.8 Extrapolation of models to N-players

Another point of interest is to understand how the traffic generated by FPS games varies as the number of players varies. We have seen that the update periods in both the client-to-server and server-to-client directions do not have a strong dependence on the number of players as they generally occur at fixed intervals. Furthermore,
2.8 Extrapolation of models to N-players

the traffic from the client to the server typically consists of small IP packets whose payload size distribution is largely independent of the number of players participating in the game [59, 102]. However, there is strong evidence that suggests that the mean server-to-client payload size increases linearly as the number of players increases [18, 60]. The work published in the literature, typically involved empirical studies of FPS games played in small test beds and the resulting traffic models have been created based on the distributions of the inter-arrival times and payload sizes for each game involving multiple players (N = 2, 3, and so on). Whilst Färber derived an extrapolation model for Half-Life Counter-Strike based on a scenario where up to 30 active players were involved, however, collection of the empirical data from games with a large numbers of players has been difficult and challenging.

2.8.1 Existing Extrapolation Methods

Beginning with some simple assumptions, Branch and Armitage [18] present a technique whereby the PMF (or PDF) of the server-to-client payload size for N-players can be predicted based on knowledge of the PMF of games with smaller numbers of players. Their assumptions are [18]:

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

Using these assumptions, they hypothesise that the random variable describing the N-player server-to-client payload can be determined by summing the random variables representing the payload size of games with fewer players. Since the PDF of the sum of two or more random variables is given by the convolution of their respective PDFs, then the PDF of an N-player game S2C payload is given by the convolution of the PDFs of the S2C payload of games with fewer players. Branch and Armitage [18] showed that this technique gives excellent agreement between the empirical and predicted distributions for the several FPS games. Branch and Armitage [18] also show that this method scales well for games with more than nine players, by arguing that the value of the $\chi^2$ discrepancy measure of the fitted model does not increase significantly as the number of players increases. To date this claim has not been supported with any empirical evidence. Although this technique gives
excellent agreement between the empirical and predicted distributions, it relies on measurements from controlled trials of games with two and three players. It is also computationally intensive, as this extrapolation technique requires repeated convolution operations. A simpler method that avoids these repeated convolutions would be advantageous for a hardware based traffic generator. This issue will be addressed further in Chapter 7.

2.9 Correlation Effects in FPS Traffic

The Autocorrelation Function (ACF) (or Autocovariance Function (ACVF)) is commonly used for investigating whether a data set is random. The ACF is usually presented as a graph that shows the autocorrelations between the data for varying time shifts (often referred to as ‘lag’). A purely random sequence of random variables will have little or no correlation with successive values of the random variable. In the case of a non-random sequence, then the correlation between successive values of this sequence will be larger than zero. Most of the current FPS traffic models are lacking, as these are based on simply drawing the random variables that represent the traffic parameters from standard distributions without considering any dependence on either the previous history or on the other participants in the game. One would expect that during the game-play phase the size of an update packet’s payload would be dependent on the level of activity at that particular time, a high level of activity would require more information per update period compared to low levels of activity. These periods of activity would last several seconds, i.e. much longer that the inter-arrival times, and so subsequent packets would have payloads of similar sizes and have some level of autocorrelation. Some authors acknowledge that correlation effects are present in FPS game traffic; for example, the client and server inter-arrivals and server packet size of Quake and Quake II exhibit significant autocorrelation [16, 85], similar results are reported by Bangun et al. [12] for Quake World. Borella attributes this correlation to be possibly due to the nature of the game and of the game play. In contrast, other authors simply ignore these effects ( [86, 88, 54, 139, 87]) as some of them claim that traffic models that incorporate correlation are more complicated than models that assume data independence [11]. It is important that this assumption be investigated if traffic models for realistic simulations are to be implemented. At this stage, there has been little work that looks at the cross-correlation in the traffic produced by the clients.
2.9 Correlation Effects in FPS Traffic

There has been some limited work that attempts to model the correlation in the traffic for FPS games. Borella [16] and Lakkakorpi et al. [85] give a measure of the correlation by calculating the autocorrelation coefficient at lag(1). Borella suggests that approaches such as the Transform Expand-Sample (TES) processes ([98]) and the Statistical Match And Queuing tool (SMAQ) [91]) could be used to incorporate the autocorrelation into traffic models. Lakkakorpi et al. [85], on the other hand, conclude that a Markov chain could be used to model the client packet sizes where different states would represent different packet sizes. Branch et al. [22] used this last approach to model the server-to-client packet size distribution for seven FPS games: Half-Life, Half-Life Counter-Strike, Quake III Arena, Quake IV, Wolfenstein Enemy Territory Pro, Half-Life 2, and Counter-Strike-Source. Branch et al. [22] claim that Markov modelling is appropriate, as the Autocorrelation Function is approximately exponential for lags up to about 500 packets, corresponding to approximately 10 to 30 seconds of game play. Although these Markov chain models capture some of the autocorrelated behaviour of the server-to-client payload size, they only generate packets whose size is an integer multiple of the median server-to-client packet size. In practice, the packet length distribution is not limited in this way.

2.9.1 Time-Series Modelling of FPS Game Traffic

Times-series models have been used extensively in the literature to model Internet traffic (e.g. Basu et al. [13]), ATM traffic (e.g. Heyman et al. [73]) as well as user behaviour in networked games ([71]). Traditional Box-Jenkins time-series [17] modelling framework comprises: Autoregressive Moving Average (ARMA) models for stationary time series and Autoregressive Integrated Moving Average (ARIMA) or Seasonal Autoregressive Integrated Moving Average (SARIMA) for non-stationary time-series. 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. The ACF, or ACVF, of a random process describes the similarity of the process with itself at different points in time (lag). Thus, the ACVF can be considered as a measure of the ‘memory’ of the process. The ACVF is useful as its behaviour can be used to identify the nature of the time-series. In general if the ACVF dies off quickly (exponentially) then the process is considered SRD, whilst if it dies off slowly (power-law) then the process is considered LRD.
2.9.2 Background on Self-similarity

Since the early 1990s, there has been great interest in describing data traffic using LRD models. The groundbreaking work of Leland, Taqqu, Willinger, and Wilson [90] in the early 90s concluded that Internet traffic showed evidence of being self-similar. A time-series exhibits LRD characteristics if its correlation structure persists over all time scales. The importance of LRD traffic is that it more heavily impacts queuing than does Poisson traffic. The Hurst parameter (H) is used to characterise LRD, a value of the Hurst parameter $\frac{1}{2} \leq H \leq 1$ indicates the presence of LRD. The methods for estimating the Hurst parameter are well covered in the literature and the reader is referred to Clegg [39] or Taqqu et al. [128] for a summary of the techniques. Recently Cevizci et al. [29] analysed the traffic characteristics for the FPS game Call of Duty 2 version 1.0 (CoD2) when connected to “middle-size servers having a maximum of 12-15 players at a time”. They have claimed that the traffic for the FPS game CoD2 has LRD characteristics in certain circumstances. If this last claim is in fact true, then FPS traffic models must not only incorporate short-term correlations, but also long term correlation effects. Cevizci et al. [29]’s claim, is based on their estimates of the Hurst parameter for three subsets of the game session. The Hurst parameter for the server-to-client packet payload size is estimated to be in the range $0.775 \leq H \leq 0.877$; whilst for the client-to-server traffic it is in the range $0.775 \leq H \leq 0.847$. Cevizci et al. [29] acknowledge that non-stationary effects will affect the estimate of Hurst parameter and for one subset of the data, they do not obtain consistent estimates of the Hurst parameter. As the game sessions were not run under controlled conditions, various factors may have led to these results, for example, different numbers of players (possibly with different abilities) participated at different times which possibly changed the nature of the game play during the session. More work with controlled experiments, such as those conducted in the SONG project [28] need to be conducted, so that this claim can be assessed further. If in fact some games do exhibit LRD behaviour, then the appropriate time-series techniques should be employed e.g. FARIMA models [62] or the more general Generalized Autoregressive Moving Average (GARMA) models [38, 63, 112].

2.10 Conclusion

In summary, much work has been done in the area of modelling FPS game traffic.
2.10 Conclusion

The mechanisms for server discovery and the traffic produced are well understood. Both the session level behaviour and start-up phase of the game have been modelled, however some discrepancies exist in the literature for the session level behaviour that need resolving.

The traffic produced in the game-play phase between the server and the client has been studied, and several simple models for particular FPS games have been proposed. One shortcoming of the work thus far is that there have been very few attempts at modelling the temporal behaviour of the FPS game traffic. In general, the models that exist in the literature are limited to modelling the IAT and packet lengths of the server-to-client and client-to-server traffic. In these models, the traffic parameters of interest are drawn from standard statistical distributions without considering any serial correlation or periodicities in the traffic.

The Extreme Value (EV) and Log-Normal (LogN) distributions have typically been chosen as models since they are able to capture the skewness in the distribution of the empirical data. However, to-date there have been few attempts at justifying why the underlying process would lead to payload lengths that are distributed in this way. In some cases, distributions whose support is the complete set of real number have been used to model the various traffic parameters without regard for the possibilities of generating parameter values that are unrealistic. An important question that remains unanswered at this stage is whether other distributions, which fit the empirical data as well, whose properties lead to better models can be found.

Some work has been done in developing techniques for extrapolating models derived from games with small numbers of players to games with larger numbers; however, these models have also neglected the temporal effects of the traffic. Furthermore, the existing extrapolation methods are computationally intensive since they are based on multiple convolution operations. Better models for extrapolating to N clients by avoiding convolution need to be developed.

The limitations of the FPS game traffic models that have been developed thus far for the game-play phase will be addressed in the following chapters of this thesis.
Chapter 3

Preliminaries and Methodology

3.1 Introduction

The goal of this research is to understand FPS game traffic and therefore develop a simple traffic model that can be used to simulate the characteristics of the traffic flowing between a FPS game server and the clients. This model should be general enough so that representative traffic can be generated for different numbers of participants and include temporal characteristics of the traffic. The model should be computationally simple, therefore techniques that avoid convolution operations, which are integral to the existing models, are desirable.

The FPS game traffic modelling in this study is based on the publicly available traffic traces for seven popular FPS games of the last decade. This chapter outlines the statistical techniques, tools, and distributions that are relevant for the analysis of the traffic traces and a description of how the fits of the statistical distributions to the models of the data can be evaluated.

The rest of the chapter is organized as follows. In Section 3.2, a brief review of some statistical terminology and basic tools is presented for the sake of completeness. Section 3.3 presents a description of statistical distributions that are typically used in modelling FPS game traffic. In Section 3.4 some useful mixture distribution are introduced along with some of their properties. Section 3.5 presents an overview of parameter estimation for fitting distributions to data, and sections 3.6 to 3.7 describe some techniques to evaluate how well the distribution fits the empirical data. Section 3.8 presents brief discussion of the ACF and the ACVF and a brief discussion of stationarity (Section 3.9). Section 3.10 and Section 3.11 present a review of time-series models. Section 3.12 shows how the traffic traces for the FPS
games that form part of this study were collected. Finally, Section 3.13 presents concluding remarks.

3.2 Tools for Stochastic Processes

In the next sections, some useful properties for the analysis of stochastic processes are outlined. These techniques will be applied in later sections of this work.

3.2.1 Distribution Function

The probability that a random variable \( X \) takes on a value less than or equal to a number \( x \) is described by the distribution function \( F_X(x) \) (or simply the distribution), equation 3.1. The distribution function is also called the CDF.

\[
F_X(x) = P[X \leq x]
\] (3.1)

3.2.2 Density Function

The density or PDF is the derivative of the CDF for a continuous random variable.

\[
f_X(x) = \frac{dF_X(x)}{dx}
\] (3.2)

The area under the PDF corresponds to the probability that the random variable lies in a specific range of values.

In the case of a discrete random variable then the relationship between the distribution function and the density function is given by:

\[
F_X(x) = \sum_{X \leq x} p(x)
\] (3.3)

In this case, \( p(x) \) is usually referred to as the PMF.

3.2.3 Sum of Random Variables

An important technique in the extrapolation of FPS traffic models to large number of players is the addition of independent random variables. Let \( X \) and \( Y \) be two random variables with PDF \( f_X(x) \) and \( f_Y(y) \), and let the random variable \( Z \) represent the sum of these random variables, i.e. \( Z = X + Y \). If \( X \) and \( Y \) are independent then for a continuous random variable, the density of \( Z \) is given by:
3.2 Tools for Stochastic Processes

\[ f_Z(z) = \int_{-\infty}^{\infty} f_Y(z-x)f_X(x)dx \]  

Equation 3.4 implies that the PDF of the sum of two independent random variables \( X \) and \( Y \) is given by the convolution of their respective PDFs [64]. This result can be expanded to sums of more than two random variables (see next section).

3.2.4 Characteristic Function

The density of the sum of two independent random variables is given by the convolution of their respective PDFs. It is well known that the convolution operation can be converted to a multiplication operation by using an appropriate transform. In the case of random variables, the characteristic function \( \varphi_X(\omega) \) [9] has this property.

The characteristic function of a real random variable \( X \) is defined as:

\[ \varphi_X(\omega) = E[e^{j\omega X}] = \int e^{j\omega x}f_X(x)dx \]  

(3.5)

Where \( f_X(x) \) is the PDF of the random variable. The characteristic function is related to the Fourier transform of the conjugate of the PDF of \( X \). The PDF can be determined from the characteristic equation by using the inversion formula.

\[ f_X(X) = \frac{1}{2\pi} \int e^{-j\omega x}\varphi_X(\omega)d\omega \]  

(3.6)

Useful properties of the characteristic function [9]:

- \( \varphi(0) = 1 \) and \( |\varphi(\omega)| \leq 1 \);
- \( \varphi(\omega) \) uniquely determines the distribution of the corresponding random variable;
- Random variables have the same distribution functions if and only if they have the same characteristic functions (Equality of distributions).

The characteristic function is useful when summing independent random variables. If \( X_1, X_2, \ldots, X_n \) are independent random variables then the characteristic function of the sum of the \( X_i \) is the product of the individual characteristic functions, i.e.:

\[ \varphi_{X_1+X_2+\ldots+X_n}(\omega) = E[e^{j\omega(X_1+X_2+\ldots+X_n)}] \]  

(3.7)

As \( X_1, X_2, \ldots, X_n \) are independent:
3 Preliminaries and Methodology

\[ \varphi_{X_1 + X_2 + \ldots + X_n}(\omega) = E[e^{j\omega X_1}] \prod E[e^{j\omega X_i}] \] (3.8)

\[ \varphi_{X_1 + X_2 + \ldots + X_n}(\omega) = \varphi_{X_1}(\omega) \varphi_{X_2}(\omega) \ldots \varphi_{X_n}(\omega) \] (3.9)

### 3.2.4.1 Transformation of Scale and Location

Consider a random variable \( X \) with characteristic function \( \varphi_X(\omega) \), then a transformation of the scale and location \((a + bX)\) results in a characteristic function \( \varphi_{(a + bX)}(\omega) \) given by:

\[ \varphi_{(a + bX)}(\omega) = E[e^{j\omega (a + bX)}] = E[e^{j\omega a} e^{j\omega bX}] = e^{j\omega a} E[e^{j\omega bX}] \] (3.10)

\[ = e^{j\omega a} \varphi_X(b\omega) \] (3.11)

### 3.2.4.2 Linear Combination of Two Random Variables

The use of Characteristic Functions to sum independent variables can be generalized for a linear combination of random variables. Assuming two independent random variables \( X \) and \( Y \) with characteristic functions \( \varphi_X(\omega) \) and \( \varphi_Y(\omega) \) respectively, the linear combination \((aX + bY)\) is given by:

\[ \varphi_{(aX + bY)}(\omega) = E[e^{j\omega (aX + bY)}] = E[e^{j\omega aX} e^{j\omega bY}] = E[e^{j\omega aX}] E[e^{j\omega bY}] \] (3.12)

\[ \varphi_{(aX + bY)}(\omega) = \varphi_X(a\omega) \varphi_Y(b\omega) \quad a, b > 0 \] (3.13)

### 3.2.4.3 Difference of Two Random Variables

The difference of random variables can also be determined by using the characteristic function. Assuming two independent random variables \( X \) and \( Y \) with characteristic functions \( \varphi_X(\omega) \) and \( \varphi_Y(\omega) \) respectively, the difference \((aX - bY)\) is given by:
\[ \varphi_{(aX-bY)}(\omega) = E \left[ e^{j\omega(aX-bY)} \right] = E \left[ e^{j\omega aX} e^{-j\omega bY} \right] = E \left[ e^{j\omega aX} \right] E \left[ e^{-j\omega bY} \right] \] (3.14)

\[ \varphi_{(aX-bY)}(\omega) = \varphi_X(a\omega)\varphi_Y(-b\omega) \quad a, b > 0 \] (3.15)

Characteristic Functions will be used in the subsequent sections when we present techniques for extrapolating FPS traffic.

### 3.3 Standard Distributions

Several distributions, described in the literature, have characteristics that are amenable to describing the shape of the payload size distribution. The payload size of the update packet by nature is discrete; but several authors model the density by a continuous function (PDF) rather than a discrete one (PMF). Normally this is not a problem, as it is understood that the random variable describing the packet size would need to be converted to an integer value, such as in the case of building a simulator. In this thesis, for simplicity, the terms CDF or distribution and PDF or density will be used irrespective of whether the distribution is discrete or continuous.

Common distributions that have been used to model the packet payload size are the Gaussian (normal), Log-Normal, and Extreme Value distributions. The Log-Normal and Extreme Value distributions are skewed, whilst the Gaussian is symmetrical. Figure 3.1 shows a comparison of the distributions. As the support for these distributions extends over ranges that are larger than what would be reasonable for realistic packet lengths, truncated versions of the distributions must be employed when these are used as part of simulations.

In the next sections, a brief review of the common distributions that have been used for modelling in FPS game traffic is presented.

#### 3.3.1 Degenerate Distribution

A degenerate or deterministic distribution [9, p. 39] is the distribution of a discrete random variable that has a single valued support \( x_0 \). The density of the degenerate distribution is given by:
The mean of the degenerate distribution is $x_0$ and the variance is zero. This distribution is useful when modelling known events, such as periodic components in the data. The distribution function for a discrete random variable can be expressed as a weighted sum of degenerate distributions. In this case:

$$F_X(x) = \sum_{X \leq x} p(X = x_i) f(x - x_i) \quad i = 1, 2, \ldots, n$$

Where $x_i$ are the possible values of the random variable, $p$ is their associated probability and $f(x - x_i)$ are degenerate distributions (impulse functions).

### 3.3.2 Gaussian Distribution

In practice, the sum of a large number of random variables will be approximately Gaussian distributed (Central Limit Theorem), thus many naturally occurring phenomena follow Gaussian distributions. The Gaussian distribution has been used to model the client-to-server packet size for Quake III [102]. The Gaussian distribution [9] is a continuous function and its PDF is defined in equation 3.18.

$$f(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$
3.3 Standard Distributions

In equation 3.18, \( \mu \) and \( \sigma \) are the mean and standard deviation of the distribution. Both the excess kurtosis and the skewness of this distribution are zero. The support for this distribution is \( x \in \mathbb{R} \) if \( \sigma^2 > 0 \). As the support for this distribution extends over the complete set of real numbers, it must be truncated if realistic packet lengths are to be simulated.

The characteristic function of the Gaussian distribution is given by:

\[
\varphi_X(\omega) = e^{(j\mu\omega - \frac{\sigma^2\omega^2}{2})} \quad (3.19)
\]

A useful property of the Gaussian distribution is that it is stable; this implies that the Gaussian distribution is closed under addition [127, p. 639], i.e. the sum of Gaussian distributed random variables is also Gaussian distributed.

3.3.3 Log-Normal Distribution

The distribution of a random variable whose logarithm is Gaussian distributed has a Log-Normal distribution [9]. This distribution has been used to model the server-to-client packet size for Quake III Arena [88] and Half-Life [87]. The Log-Normal distribution is a positively skewed continuous function and its PDF is given in equation 3.20:

\[
f(x, \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(\frac{-(\ln(x) - \mu)^2}{2\sigma^2}\right) \quad (3.20)
\]

In this case, \( \mu \) and \( \sigma \) are the location and scale parameters of the distribution. The mean and variance of this distribution are given by:

\[
E(X) = \exp(\mu + \frac{\sigma^2}{2}) \quad (3.21)
\]

\[
Var(X) = (e^{\sigma^2} - 1) \exp(2\mu + \sigma^2) \quad (3.22)
\]

The support for the Log-Normal distribution is \( x \in (0, +\infty) \) and the skewness and excess kurtosis of this distribution are given by equations 3.23 and 3.24 respectively.

\[
skewness = (e^{\sigma^2} + 2)\sqrt{e^{\sigma^2} - 1} \quad (3.23)
\]

\[
kurtosis = e^{4\sigma^2} + 2e^{3\sigma^2} + 3e^{2\sigma^2} - 3 \quad (3.24)
\]
3.3.4 Generalised Extreme Value Distribution

The Extreme Value distribution is one of a family of distributions that are used to model events, such as extreme weather conditions. The three common forms of Extreme Value distributions are the Type I or Gumbel Type distribution, the Type II or Fréchet Type distribution and the Type III or Weibull Type distribution [9]. Of the three types, the Gumbel distribution has been used extensively to model the packet size distributions for various FPS games. For example, Quake I [16], and Quake II [85], Half-Life Counter Strike [59], Halo II [139] and the IEEE 802.16m FPS game traffic model [129]. The Gumbel distribution is a positively skewed continuous function and its PDF is defined as:

\[
f(x, \mu, \sigma) = \frac{1}{\sigma} \exp\left(-\frac{x - \mu}{\sigma}\right) \exp\left(-e^{-\frac{x - \mu}{\sigma}}\right)
\]  

(3.25)

Where \( \mu \) and \( \sigma \) are the location and scale parameters of the distribution. The mean and variance of the Gumbel distribution are given by equations 3.26 and 3.27:

\[
E(X) \approx \mu + 0.57721\sigma
\]  

(3.26)

\[
Var(X) = \frac{\pi^2}{6} \sigma^2
\]  

(3.27)

The support for the Gumbel distribution is \( x \in \mathbb{R} \). Again care must be taken when using this distribution so as not to generate negative quantities. The skewness and kurtosis of this distribution are approximately 1.14 and 2.4 respectively.

The Gumbel distribution is a special case of the Fisher-Tippett distribution, which is commonly known as the Generalised Extreme Value (GEV) distribution. The PDF for this distribution is given by equation 3.28.

\[
f(x, \mu, \sigma, \gamma) = \begin{cases} 
\frac{1}{\sigma} \left(1 + \gamma \left(\frac{x - \mu}{\sigma}\right)\right)^{-\frac{1}{\gamma} - 1} \exp\left(-\left(1 + \gamma \left(\frac{x - \mu}{\sigma}\right)\right)^{-\frac{1}{\gamma}}\right) & \gamma \neq 0 \\
\frac{1}{\sigma} \exp\left(-\frac{x - \mu}{\sigma}\right) \exp\left(-e^{-\frac{x - \mu}{\sigma}}\right) & \gamma = 0
\end{cases}
\]  

(3.28)

The value of the shape parameter \( \gamma \) determines type of Extreme Value distribution. The case where \( \gamma \to 0 \) corresponds to the Gumbel distribution discussed above. For the case where the shape parameter \( \gamma = 0 \), the support is the same as the Gumbel distribution \( x \in \mathbb{R} \), whilst for \( \gamma > 0 \) (Fréchet) the support is \( x \in [\mu - \frac{\sigma}{\gamma}, +\infty) \).
3.3 Standard Distributions

and for $\gamma < 0$ (Weibull) the support is $x \in (-\infty, \mu - \frac{\sigma}{\gamma}]$. The mean and variance of the GEV distribution are given by equations 3.29 and 3.30 respectively.

$$E(X) = \begin{cases} 
\mu + \sigma \Gamma(1-\gamma) & \gamma \neq 0 \quad \gamma < 1 \\
\mu + 0.57721\sigma & \gamma = 0 \\
does \ not \ exist & \gamma \geq 1 
\end{cases}$$ (3.29)

$$Var(X) = \begin{cases} 
\frac{\sigma^2(\alpha_2 - \alpha_1)}{\gamma^2\sigma^2} & \gamma \neq 0 \quad \gamma < 1 \\
\frac{\pi^2}{6} & \gamma = 0 \\
does \ not \ exist & \gamma \geq \frac{1}{2} 
\end{cases}$$ (3.30)

$$\alpha_k = \Gamma(1 - k\gamma) \quad k = 1, 2, 3, 4$$ (3.31)

The $k^{th}$ moment ($\alpha_k$) of this distribution is given by equation 3.31. These moments can then be used to determine the excess kurtosis and skewness by using equations 3.32 and 3.33.

$$skewness = \frac{3\alpha_2\alpha_1 - \alpha_3 - 2\alpha_1^2}{(\alpha_2 - 2\alpha_1^2)^{\frac{3}{2}}}$$ (3.32)

$$kurtosis = \frac{\alpha_4 - 4\alpha_1\alpha_3 + 6\alpha_2\alpha_1^2 - 2\alpha_1^4}{(\alpha_2 - \alpha_1^2)^2}$$ (3.33)

As previously stated, many authors use the Extreme Value distribution (Gumbel) to model the payload size of FPS games. A restriction of this approach that the Gumbel distribution has fixed skewness and excess kurtosis and that it lacks a ‘shape’ parameter. To overcome these restrictions the GEV Distribution will be used in this work rather than the Gumbel distribution.

3.3.5 Gamma Distribution

As far as the author is aware, the Gamma distribution has not been used to model FPS traffic (besides the exponential distribution). However, it is frequently used to model waiting times and lifetimes. The Gamma distribution [9, p179] is also a positively skewed continuous function and its PDF is defined as:

$$f(x, \mu, \sigma, \gamma) = \frac{(x-\mu)^{\gamma-1}e^{-\frac{x-\mu}{\sigma}}}{\Gamma(\gamma)\sigma^\gamma} \quad x > \mu \quad \gamma, \sigma > 0$$ (3.34)
Where $\gamma$ is the shape parameter, $\mu$ the location parameter, $\sigma$ the scale parameter and $\Gamma(\cdot)$ is the gamma function. The mean and variance of the Gamma distribution are given by equations 3.35 and 3.36:

$$E(X) = \mu + \gamma \sigma$$ \hspace{1cm} (3.35)

$$Var(X) = \gamma \sigma^2$$ \hspace{1cm} (3.36)

The support for the Gamma distribution is $x \in [\mu, \infty)$. The skewness and kurtosis of this distribution are $\frac{\gamma}{\sqrt{\gamma}}$ and $\frac{6}{\gamma}$ respectively. The standard Gamma distribution is commonly used in the literature, in this case the location parameter $\mu = 0$, and so the density becomes:

$$f(x, \sigma, \gamma) = \frac{x^{\gamma-1}}{\Gamma(\gamma)\sigma^\gamma} e^{-\frac{x}{\sigma}}, \hspace{0.5cm} x \geq 0, \hspace{0.5cm} \gamma, \sigma > 0$$ \hspace{1cm} (3.37)

The characteristic function of the standard Gamma distribution is given by:

$$\varphi_X(\omega) = (1 - j\sigma\omega)^{-\gamma}$$ \hspace{1cm} (3.38)

The Gamma distribution is also closed under addition, in the case where the random variables are Gamma distributed with the same scale parameter.

The Exponential distribution is a special case of the Gamma distribution with the shape parameter $\gamma = 1$ and $\lambda = \frac{1}{\sigma}$. In this case, the density becomes:

$$f(x, \lambda) = \lambda e^{-\lambda x}, \hspace{0.5cm} x \geq 0$$ \hspace{1cm} (3.39)

3.4 Mixture Distributions

Mixture distributions arise from functions of random variables with different distributions. Mixtures of random variables have been used in modelling some aspects of the traffic in FPS games. Lang et al. [88] use a mixture of degenerate and exponential distributions to model the IAT of Quake III. In the following sections, a number of mixture distributions that we will use in modelling and extrapolating FPS game traffic are presented.
3.4 Mixture Distributions

3.4.1 Ex-Gaussian Distribution

The Ex-Gaussian (EG) [93] (or Exponentially Modified Gaussian) distribution is a skewed mixture distribution that has been used to model reaction times. This distribution has not been used for modelling the packet size distribution of FPS traffic; however, it has characteristics that are similar to that of the Log-Normal and Gumbel distributions detailed above. The EG distribution arises from the sum of two random variables, one of which is Gaussian distributed \( \text{mean} = \mu, \text{st. dev} = \sigma \) and the other exponentially distributed \( \text{rate} = \lambda \). The PDF of the EG distribution is given by:

\[
f(x, \mu, \sigma, \lambda) = \lambda \exp \left( (\mu - x) \lambda + \frac{\lambda^2 \sigma^2}{2} \right) \sqrt{\frac{2}{\pi}} \exp \left( \frac{x - \mu - \lambda \sigma}{\sigma} \right) \Theta \left( x - \mu - \lambda \sigma \right) \quad \sigma > 0
\]  

(3.40)

Where \( \Theta(\cdot) \) is the standard Gaussian cumulative distribution, refer to Appendix C section C.2 for the derivation. The mean and variance of this distribution are given by:

\[
E(X) = \mu + \frac{1}{\lambda}
\]  

(3.41)

\[
\text{Var}(X) = \sigma^2 + \frac{1}{\lambda^2}
\]  

(3.42)

The support for the EG distribution is \( x \in \mathbb{R} \) and the skewness and excess kurtosis of this distribution are given in equations 3.43 and 3.44. Again, since the support for this distribution extends over the complete set of real numbers, it must be truncated if realistic packet lengths are to be simulated.

\[
\text{skewness} = \frac{2\lambda^3}{(1 + \lambda^2 \sigma^2)^{1/2}}
\]  

(3.43)

\[
\text{kurtosis} = \frac{6}{(1 + \lambda^2 \sigma^2)^2}
\]  

(3.44)

The characteristic function of the EG distribution is:

\[
\varphi(\omega) = \frac{\exp \left( j\mu \omega - \frac{\sigma^2 \omega^2}{2} \right)}{1 - \frac{j\omega}{\lambda}}
\]  

(3.45)
3.4.2 Gamma Modified Gaussian Distribution

The Gamma Modified Gaussian (ΓMG) distribution is a skewed mixture distribution that to date has not been used for modelling the packet size distribution of FPS traffic; however, it can be thought of as a generalisation of the EG distribution described in the previous section, and so this distribution may be useful in this context. The ΓMG distribution arises from the sum of two random variables, one of which is Gaussian distributed \((\text{mean} = \mu, \text{st.dev} = \sigma)\) and the other Gamma distributed \((\text{shape} = \gamma, \text{scale} = \kappa)\). The PDF of the ΓMG (equation 3.46) can be derived from solving the convolution of the Gamma and Gaussian densities. However, the expression is complicated. The reader is referred to Appendix C.1 for the derivation.

\[
g(u; k, \theta, \sigma, \mu) = \frac{2^{k-3}}{\theta^k} \sigma^{1-k} e^{\left(\frac{u^2}{2\theta^2} - \frac{(u-\mu)^2}{\theta^2}\right)} I_{k-1} \text{erfc}\left(\frac{\sigma^2 - (u-\mu)}{\sqrt{2}\sigma\theta}\right) \tag{3.46}
\]

The characteristic function of the ΓMG distribution is given by the product of the characteristic functions of the Gaussian and Gamma distributions.

\[
\varphi(\omega) = (1 - j\kappa \omega)^{-\gamma} \exp\left(j \mu \omega - \frac{\sigma^2 \omega^2}{2}\right) \tag{3.47}
\]

The mean and variance of this distribution are given by:

\[
E(X) = \mu + \gamma \kappa \tag{3.48}
\]

\[
Var(X) = \sigma^2 + \gamma \kappa^2 \tag{3.49}
\]

The support for the ΓMG distribution is \(x \in \mathbb{R}\) and the skewness and excess kurtosis of this distribution are given in equations 3.50 and 3.51.

\[
skewness = \frac{2\gamma \kappa^3}{(\sigma^2 + \gamma \kappa^2)^{3/2}} \tag{3.50}
\]

\[
kurtosis = \frac{6\gamma \kappa^4}{(\sigma^2 + \gamma \kappa^2)^2} \tag{3.51}
\]
3.4.3 Linear combination of Gamma Modified Gaussian distributed variables

In Section 3.2.4.2 it was shown that the characteristic function of a linear combination of random variables \( aX + bY \) is given by equation 3.13, i.e. the product of the respective scaled characteristic functions. ‘Decomposition’ is the process of expressing a random variable as the sum of two or more independent random variables. Consider two independent and identically distributed (iid) ΓMG random variables \( X \) and \( Y \) with the same scale parameter \((\kappa)\), and with characteristic functions given by:

\[
\varphi_X(\omega) = (1 - j\kappa\omega)^{-\gamma_X} \exp\left(j\mu_X\omega - \frac{\sigma_X^2\omega^2}{2}\right)
\]

\[
\varphi_Y(\omega) = (1 - j\kappa\omega)^{-\gamma_Y} \exp\left(j\mu_Y\omega - \frac{\sigma_Y^2\omega^2}{2}\right)
\]

Using the result of equation 3.13, the sum of two iid ΓMG random variables \( X \) and \( Y \) is given by:

\[
\varphi_{X+Y}(\omega) = (1 - j\kappa\omega)^{-(\gamma_X+\gamma_Y)} \exp\left[j(\mu_X + \mu_Y)\omega - \frac{(\sigma_X^2 + \sigma_Y^2)\omega^2}{2}\right]
\]

This is also the characteristic function of a ΓMG distribution. This result implies that the sum of ΓMG distributed random variables, with the same scale parameter \((\kappa)\), is also ΓMG distributed, which in turn implies that the ΓMG is closed under addition.

The \( \gamma, \mu \) and \( \sigma \) parameters of this sum are:

\[
\mu_{X+Y} = \mu_X + \mu_Y, \quad \sigma_{X+Y}^2 = \sigma_X^2 + \sigma_Y^2 \quad \gamma_{X+Y} = \gamma_X + \gamma_Y
\]

These results are useful, as they imply that the ΓMG distribution can be decomposed into the sum of independent random variables that follow a ΓMG distribution. (Subject to these having the same scale parameter.)

In the case where \( \gamma_X = \gamma_Y = \frac{1}{2}, \mu_X = \mu_Y = \frac{\mu}{2}, \sigma_X^2 = \sigma_Y^2 = \frac{\sigma^2}{2} \) equation 3.53 becomes:
\[ \varphi (\omega ) = (1 - j \kappa \omega )^{-1} \exp \left( j \mu \omega - \frac{\sigma^2 \omega^2}{2} \right) \quad (3.55) \]

This expression is equivalent to equation 3.45 with \( \kappa = \frac{1}{\lambda} \), so by the equivalence of characteristic functions (3.2.4), this distribution is the EG distribution. This result implies that the decomposition of the EG distribution is a \( \Gamma_{MG} \) distribution with parameters \( \gamma_{GMG} = \frac{1}{2} \), \( \mu_{GMG} = \frac{\mu_{EG}}{2} \), \( \sigma^2_{GMG} = \frac{\sigma^2_{EG}}{2} \).

The difference of two iid \( \Gamma_{MG} \) random variables \( X \) and \( Y \) with the same \( \kappa \) and \( \gamma \) can be determined using equation 3.15:

\[ \varphi_{X - bY} (\omega) = (1 - j \kappa \omega)^{-\gamma} (1 + jb \kappa \omega)^{-\gamma} \exp \left[ j (\mu_X - b \mu_Y) \omega - \frac{(\sigma^2_X + b^2 \sigma^2_Y) \omega^2}{2} \right] \quad (3.56) \]

\[ \varphi_{X - bY} (\omega) = \left[ 1 + b \kappa^2 \omega^2 - j \kappa \omega (1 - b) \right]^{-\gamma} \exp \left[ j (\mu_X - b \mu_Y) \omega - \frac{(\sigma^2_X + b^2 \sigma^2_Y) \omega^2}{2} \right] \quad (3.57) \]

Equation 3.57 is the product of a Gaussian characteristic function and a
\[ \left[ (1 + b \kappa^2 \omega^2 - j \kappa \omega (1 - b) \right]^{-\gamma} \] term, which is the characteristic function of a Variance Gamma distribution [94]. Therefore, the density of the difference of two Gamma Modified Gaussian distributed random variables is given by the convolution of a Variance Gamma and a Gaussian density or alternatively by the sum of a Variance Gamma (VT) and a Gaussian distributed random variable. This distribution will be referred to as the Variance Gamma Modified Gaussian (VTG) distribution in this work.

If \( \gamma_X = \gamma_Y = \gamma, \mu_X = \mu_Y = \mu, \sigma^2_X = \sigma^2_Y = \sigma^2 \) the equation 3.57 becomes the characteristic function of the VTG distribution:

\[ \varphi (\omega) = \left[ 1 + b \kappa^2 \omega^2 - j \kappa \omega (1 - b) \right]^{-\gamma} \exp \left[ j \mu (1 - b) \omega - \frac{\sigma^2 (1 + b^2) \omega^2}{2} \right] \quad (3.58) \]

The sum of two iid VTG random variables with the same \( b \) and \( \kappa \) is:

\[ \varphi_{X + Y} (\omega) = \left[ 1 + b \kappa^2 \omega^2 - j \kappa \omega (1 - b) \right]^{-\gamma_X} \exp \left[ j \mu_X (1 - b) \omega - \frac{\sigma^2_X (1 + b^2) \omega^2}{2} \right] + \]
\[ + \left[ 1 + bk^2\omega^2 - j\kappa\omega(1 - b) \right]^{-\gamma} \exp \left[ j\mu_Y(1 - b)\omega - \frac{\sigma_Y^2(1 + b^2)\omega^2}{2} \right] \]  

\[ = \left[ 1 + bk^2\omega^2 - j\kappa\omega(1 - b) \right]^{-\gamma} \nu \exp \left[ j(\mu_X + \mu_Y)(1 - b)\omega - \frac{(\sigma_X^2 + \sigma_Y^2)(1 + b^2)\omega^2}{2} \right] \]  

This is also the characteristic function of a VΓG distribution, thus under these conditions, the VΓG is closed under addition.

The mean and variance of the VΓG distribution are:

\[ E(X) = (1 - b)(\gamma\kappa + \mu) \]  

\[ Var(X) = (1 + b^2)(\sigma^2 + \gamma\kappa^2) \]  

The support for the VΓG distribution is \( x \in \mathbb{R} \) and the skewness and excess kurtosis of this distribution are given in equations 3.63 and 3.64.

\[ \text{skewness} = \frac{-2(b^3 - 1)\gamma\kappa^3}{[(1 + b^2)(\sigma^2 + \gamma\kappa^2)]^{\frac{3}{2}}} \]  

\[ \text{kurtosis} = \frac{6(b^4 + 1)\gamma\kappa^4}{[(1 + b^2)(\sigma^2 + \gamma\kappa^2)]^2} \]  

In the previous two sections, we have introduced two novel mixture distributions and outlined their statistical properties. These distributions will be used in modelling the FPS traffic in Chapter 7.

### 3.5 Parameter Estimation

Once a particular statistical distribution has been chosen as the model, its parameters need to be estimated in order to fit it to the observed data. Two common methods that are available, the Method of Moments (MM) and the Maximum Likelihood Estimation (MLE), are described in the following sections.
3.5.1 Method of Moments

The Method of Moments [97] estimates the parameters of a distribution by matching the sample moments (from the sample data) to the population moments.

The moments ($\mu_n$) for a particular distribution can be determined using the characteristic function:

$$\mu_n = E(X^n) = j^{-n} \left. \frac{d^n}{d\omega^n} \phi_X(\omega) \right|_{\omega=0} \quad (3.65)$$

Assuming that a distribution has $k$ parameters $\{\alpha_1, \alpha_2, \ldots \alpha_k\}$, these parameters can be estimated as follows (from [97]):

- express the $k$ parameters as functions of the first $k$ moments of the distribution
  $$\alpha_1 = f_1 \{\mu_1, \mu_2, \ldots, \mu_k\}, \alpha_2 = f_2 \{\mu_1, \mu_2, \ldots, \mu_k\} \ldots \alpha_k = f_k \{\mu_1, \mu_2, \ldots, \mu_k\}$$
- calculate the first $k$ sample moments from the empirical data and substitute these into the above functions.
- solve for the estimated parameters using the following:
  $$\hat{\alpha}_1 = f_1 \{\bar{X}, \bar{X}^2, \ldots, \bar{X}^k\}, \hat{\alpha}_2 = f_2 \{\bar{X}, \bar{X}^2, \ldots, \bar{X}^k\} \ldots \hat{\alpha}_k = f_k \{\bar{X}, \bar{X}^2, \ldots, \bar{X}^k\}$$

The advantage of using the MM is that the parameters can be estimated without knowing the PDF of the distribution. This method can also be computationally simple.

3.5.2 Maximum Likelihood Estimation

Maximum Likelihood Estimation on the other hand, is a robust parameter estimation method that is used to determine the statistical parameters that result in the empirical data’s likelihood function being maximised. The parameters of a PDF can be determined by using the following (from [117]):

Assuming that $x$ is a continuous random variable with PDF:

$$f(x; \theta_1, \theta_2, \theta_3 \ldots \theta_n) \quad (3.66)$$

here $\theta_1, \theta_2, \theta_3 \ldots \theta_n$ are the $n$ unknown parameters that need estimating.

The likelihood function $L$ for $N$ independent observations of $x$ can be expressed by the product:

$$L(x_1, x_2, \ldots, x_N|\theta_1, \theta_2, \theta_3 \ldots \theta_n) = \prod_{i=1}^{N} f(x_i; \theta_1, \theta_2, \theta_3 \ldots \theta_n) \quad (3.67)$$
3.6 Quantile-Quantile Plots

As it is more convenient to work with summations rather than products, the logarithm of the likelihood function is used:

\[ \Lambda = \sum_{i=1}^{N} \ln( f(x_i; \theta_1, \theta_2, \theta_3 \ldots \theta_n) ) \]  

(3.68)

The maximum likelihood estimators of the parameters \( \theta_1, \theta_2, \theta_3 \ldots \theta_n \) can be found from maximising either equation 3.67 or equation 3.68. The solution is given by:

\[ \frac{\partial \Lambda}{\partial \theta_i} = 0, \ j = 1, 2, \ldots, n \]  

(3.69)

In this thesis, the Matlab Statistics Toolbox was used to fit distributions using the MLE method. In cases where the MLE method for a particular distribution was not implemented in Matlab, the details of how the distribution was fitted are provided.

3.6 Quantile-Quantile Plots

Two distributions can be compared to each other by plotting the fraction of the values that lie below a given point (quantiles) of the two distributions against each other. This is called a Q-Q plot and it is useful as a qualitative measure to determine whether two data-sets come from populations with a common distribution (see Figure 3.2). When the resulting points of the Q-Q plot fall roughly on a 45-degree reference line, then the data-sets most likely have a common distribution. A departure from this line signifies that the two data-sets may have different distributions. The sample sizes of the two data-sets need not be equal for the comparison to be meaningful. The quantiles of an empirical data-set can be compared to those of a standard distribution in order to determine if these observations belong to that distribution. Another useful feature of the Q-Q plot is that it can also provide an indication of where a data-set deviates from a particular distribution, for example deviations in the tails or in the main body of the distribution. Furthermore, the Q-Q plot can reveal shifts in either the scale and location parameters as well as differences in the symmetry between two distributions.
3 Preliminaries and Methodology

The Q-Q plots introduced in the previous section are useful as a visual aid to help determine the best-fit distribution for a particular data-set. However, in order to evaluate how well a model agrees with a set of empirical observations, some form of goodness-of-fit metric can be used. Borella [16], Lakkakorpi et al. [85], and others suggest the use of the $\lambda^2$ (proposed by Pederson and Johnson [107]) based on work by Paxson [104] for the modelling of network traffic data. The $\lambda^2$ discrepancy measure gives an indication of how closely a distribution fits the observed data. The lower the $\lambda^2$ discrepancy measure the better the fit. Unlike the traditional goodness-of-fit tests such as the Chi-square ($\chi^2$), Anderson-Darling and K-S tests that are used to test the hypothesis that the data comes from a particular distribution, the $\lambda^2$ discrepancy measure is less sensitive to the number of bins and the standard deviation of the data [104]. It is also claimed that the traditional goodness-of-fit tests, often do not work well with large data sets that are autocorrelated [16]. The FPS game traffic traces have a very large number of packets that are highly correlated, thus the use of the $\lambda^2$ over the traditional goodness-of-fit tests is warranted.

The $\lambda^2$ discrepancy measure is defined as (see [16]):
\[ \chi^2 = \frac{\chi^2 - K - df}{n - 1} \]  

(3.70)

Where \( n \) is the number of observations in the dataset. \( \chi^2 \) is a chi-square goodness of fit test as defined by equation 3.71, \( df \) (the number of degrees of freedom of the \( \chi^2 \) test) is the number of \( \chi^2 \) bins minus one, that are used in the test, minus the number of parameters used to estimate the analytical distribution.

\[ \chi^2 = \sum_i^N \frac{(E_i - T_i)^2}{T_i} \]  

(3.71)

\( N \) is the number of bins with width \( w \), given by:

\[ w = 3.49\sigma n^{-\frac{1}{3}} \]  

(3.72)

where \( \sigma \) is the standard deviation of the empirical data.

\( E_i \) is the number of items in the \( i^{th} \) bin of the empirical data set and \( T_i \) is the number of items in the \( i^{th} \) bin of the reference distribution.

\( K \) is given by:

\[ K = \sum_i^N \frac{E_i - T_i}{T_i} \]  

(3.73)

In the case where \( T_i = 0 \) (degenerate distributions) a modified version of the \( \chi^2 \) test is required to avoid divide by zero errors.

\[ \chi^2 = \sum_i^N \frac{(T_i - E_i)^2}{E_i} \]  

(3.74)

For this case

\[ K = \sum_i^N \frac{T_i - E_i}{E_i} \]  

(3.75)

### 3.8 Autocovariance and Autocorrelation Functions

The ACF and ACVF of a random process give an indication of the extent that the present observation in a time-series is similar to the previous one and so they are commonly used in investigating the randomness or otherwise of a time-series. The terms autocovariance and autocorrelation are often used interchangeably. As these functions, describe the similarity of a process with itself for varying time
shorts at different points in time often referred to as ‘lag’. This term is not to be confused with the latency that a player experiences in an on-line game which is also referred to as ‘lag’. A purely random sequence of observations will exhibit near zero correlation with successive observations, whereas for a correlated process the correlation between successive observations will be significantly greater than zero.

The ACVF of a stochastic process $X(t)$ is defined as:

$$\gamma(k) = \text{Cov}(X(t), X(t+k)) = E[(X(t) - E[X(t)])(X(t+k) - E[X(t)])]$$  \hspace{1cm} (3.76)

$$\gamma(k) = E[(X(t) - \mu)(X(t+k) - \mu)]$$  \hspace{1cm} (3.77)

Where $\mu$ is the mean of the process. The ACF on the other hand is the scale normalized ACVF and is obtained by dividing the Autocovariance Function by the variance $\sigma^2$ of the process.

$$\rho(k) = \frac{\gamma(k)}{\sigma^2} = \frac{\gamma(k)}{\gamma(0)}$$  \hspace{1cm} (3.78)

Both the ACVF and ACF can be considered as being a measure of the ‘memory’ of the process. Typically, if the ACVF dies off quickly then the process is usually stationary.

A related concept is that of a Partial Autocorrelation Function (PACF) [17]. The PACF measures the correlation between $x(t)$ and $x(t-k)$ when the intervening correlations of other lags $x(t-1)$ and $x(t-k-1)$ have been removed. The PACF is commonly used to identify an appropriate Box-Jenkins time-series model.

3.9 Stationarity

The issue of stationarity is important when dealing with time-series models. Box et al. [17] discuss this issue in detail. The following is a summary of stationarity issues taken largely from Box et al. [17]. Stationarity implies that no statistical properties of a random process vary with time. Box et al. [17] refer to this as a state of statistical equilibrium. The strictest form of stationarity (strict sense stationary) requires that the joint probability distribution of a set of observations is not affected by the time at which the observations were made. In this case the joint probability distribution of set of observations $X(t) = x_{t1}, x_{t2}, ..., x_{tn}$ taken at times $t_1, t_2, t_3, t_n$
and that of \( X(t + \tau) = x_{t_1 + \tau}, x_{t_2 + \tau}, \ldots, x_{t_n + \tau} \) taken at times \( t_1 + \tau, t_2 + \tau, t_3 + \tau, \ldots, t_n + \tau \) are the same. This implies that the joint probability distribution is completely independent of \( t \) and thus any time shift \( \tau \). This condition implies that all the higher-order moments are also independent of time. A less stringent form of stationarity is wide sense stationarity (covariance stationarity). In this case the mean and variance are constant and do not depend on any time shift, whilst the ACVF between any two observations depends only on the time difference between the two rather than their individual locations. This last statement implies that the ACVF has the form shown in equation 3.79:

\[
\text{Cov}[X(t - j), X(t - j - \tau)] = E[X(t)X(t - \tau)] = E[X(t - j)X(t - j - \tau)]
\]

Stationarity can be assessed from a runs test (non-parametric test) or from the ACVF plot (parametric test). According to Box et al. [17], non-stationarity is usually indicated by an ACVF plot that does not cut off nor decay exponentially.

Some causes of non-stationarity in time series are stochastic drift and seasonal variations. Stochastic drift refers to situations where the average value of the process changes with time. In general, the drift component is not periodic and is usually called a trend. Classical decomposition [24, p. 14] can be used to identify the trend in a time-series. Trends can also be removed by differencing. In the cases where the time-series varies around the mean in a cyclical fashion, the time-series has a seasonal dependency (seasonality) and so is not stationary. Seasonality can be determined visually by observing either a plot of the ACVF (autocorrelogram) or the spectrum of the time-series.

In Box-Jenkins analysis, when a time series is not stationary, then differencing of some form may give rise to a stationary time series. Differencing involves the modelling of the differences between consecutive observations.

### 3.10 Time-series Models

Time-series modelling attempts to describe the underlying nature of a process, which is represented by a sequence of observations, in order to forecast its future values. Generally, for a time-series the observations occur at fixed regular intervals. In a FPS game, under ideal conditions, a server would send update packets at a defined periodic rate, implying that the Inter-arrival time (IAT) of the update packets would
be deterministic with constant intervals between updates. However, in practice the measured IATs also have a random component that is due to various delays, such as those in the network. In this work, we are interested in the modelling the payload size of the traffic and not the IATs. Therefore, the traces will be parameterised by lag \((n)\).

### 3.10.1 Parsimony

When using time-series models it is desirable to find the simplest models that provide an adequate description of the observed data. This is known as the principle of parsimony [17, p. 81].

When choosing suitable models for a time-series one needs to be careful not to over-fit the model to the data. Over-fitting [35, p. 79] will result in a model that gives a more accurate match to the particular measured data for the traffic; however, one must keep in mind that the measurements relate to only one possible realisation of the underlying random process. A simple model that will give an acceptable fit to the measured data, whilst still capturing the essential statistical features of the time-series is desirable. The principle of parsimony will be used to guide the choice of the model that is adopted. Parsimony states that the model should be chosen such that it has the minimum number of parameters that give an adequate representation of the measured data. The Akaike Information Criterion (AIC) is a measure of parsimony that penalizes models with a large number of parameters, thus using the AIC can help prevent choosing models that over-fit the data set. Box and Jenkins define the AIC as:

\[
AIC \approx \log(\hat{\sigma}^2) + \frac{r}{n}
\]

where \(\hat{\sigma}^2\) is the estimate of the variance of the process, \(r\) is the number of estimated parameters, and \(n\) is the number of observations in the data set. Another measure of parsimony that could be used is the Bayesian Information Criterion (BIC), In this work we have chosen to use the AIC as this model penalises the larger models less than does the BIC.

### 3.11 Generalized Autoregressive Moving Average Model

A general time-series model that can deal with time-series that may be stationary or non-stationary, long-range or short-range dependent or that include periodic components is the Generalized Autoregressive Moving Average (GARMA) model [38, 63, 112].

The GARMA model of a time-series \(X_n\) is defined as [112]:
\[ \Phi(B)(1 - 2\eta B + B^2)^\delta(X_n - \mu) = \Theta(B)\varepsilon_n, \quad |\delta| \leq \frac{1}{2}, \quad |\eta| \leq 1 \quad (3.80) \]

Where \( \mu \) is the mean of the process and \( \varepsilon_n \), (the residuals, innovations, or shocks) are iid random variables with zero mean and variance \( \sigma^2 \). \( \Phi(B) \) is the autoregressive polynomial of degree \( p \) given by equation 3.81 and \( \Theta(B) \) is the moving average polynomial of order \( q \) given by equation 3.82.

\[ \Phi(B) = 1 - \phi_1 B^1 - \phi_2 B^2 - \cdots - \phi_p B^p \quad (3.81) \]

\[ \Theta(B) = 1 - \theta_1 B^1 - \theta_2 B^2 - \cdots - \theta_q B^q \quad (3.82) \]

The \( \Phi(B) \) and \( \Theta(B) \) polynomials are defined in terms of a back-shift operator \( B \), which is defined as:

\[ B^i X_n = X_{n-i} \quad i = 0, \pm 1, \pm 2 \ldots \quad (3.83) \]

The term given by \( (1 - 2\eta B + B^2)^\delta \) is the Gegenbauer polynomial [100, p97], \( \delta \) is the fractional differencing parameter and \( \eta \) is the periodicity parameter, these last two parameters model the long-range and the periodic nature of the process. The Gegenbauer polynomial can be expressed as a series that is given by:

\[ (1 - 2\eta B + B^2)^{-\delta} = \sum_{k=0}^{\infty} C_k^{(\delta)}(\eta) B^k \quad (3.84) \]

where

\[ C_k^{(\delta)} = \sum_{j=0}^{\left[ \frac{k}{2} \right]} (-1)^n \frac{\Gamma(\delta+(k-j))}{\Gamma(\delta)} \frac{(2\eta)^{k-2j}}{j(k-j)!} \quad (3.85) \]

Here \( \left[ \frac{k}{2} \right] \) is the largest integer less than or equal to \( \frac{k}{2} \).

The GARMA model degenerates to an ARMA model when \( \delta = 0 \), an ARIMA model when \( \eta = 1, \delta = \frac{1}{2} \) and a FARIMA model when \( \eta = 1 \) [112].

Although the GARMA model is general, it may be difficult to implement in a simulation model as the Gegenbauer polynomial (equation 3.84) is an infinite sum. Truncating this sum leads to correlations appearing at lags that are multiples of
the length of the truncated series. Furthermore, not every time-series will be non-stationary, or have periodic components. In these cases, parsimony would dictate that a simpler model should be used. In practice, a simpler model may be adequate, as it may be possible to remove the periodic components and trends that cause the non-stationarity of the time-series.

3.11.1 Short-Range Dependent (SRD) Processes

The GARMA process, defined in equation 3.11, can be used as a general model for FPS game traffic irrespective of whether the traffic statistics are stationary or not. In essence, for the time-series to be stationary it should not have drifts or periodicities (Section 3.9). Whilst this is a general model, parsimony may lead us to choose one of the limiting cases of this model when appropriate. In the case where the time-series has an ACVF that dies off in an exponential fashion and has no trend or seasonal effect, a simpler model would be more parsimonious. These processes are termed SRD. Four examples of SRD processes are presented in the following sections.

3.11.1.1 Autoregressive Moving Average (ARMA) Process

For a stationary time-series $X_n$, both the differencing ($d$) and periodicity ($\eta$) parameters in equation 3.80 are zero. This implies that the Gegenbauer polynomial (equation 3.84) is equal to 1. In this case, the process is a Box-Jenkins Autoregressive Moving Average ARMA($p,q$) process and is defined as:

$$\Phi(B)(X_n-\mu) = \Theta(B)\varepsilon_n$$  \hspace{1cm} (3.86)

Again $B$ is the back-shift operator (3.83), $\mu$ is the mean of the process, and $\Phi(B)$ and $\Theta(B)$ are given by equations 3.81 and 3.82 respectively.

The residuals $\varepsilon_n$ in equation 3.86 are assumed to be iid random variables with zero mean and variance $\sigma^2$.

For a zero mean process, equation 3.86 simplifies to:

$$\Phi(B)X_n = \Theta(B)\varepsilon_n$$  \hspace{1cm} (3.87)

The ACF of an ARMA($p,q$) process is either a damped exponential or a damped sinusoid that decays exponentially to zero after the largest lag of the MA component [17, p. 81].
3.11 Generalized Autoregressive Moving Average Model

The ARMA(1,1) process is a special case of the more general ARMA(p,q). This process, defined by equation 3.88, will be used later in this work,

\[ X_n - \phi_1 X_{n-1} - \mu = \varepsilon_n - \theta_1 \varepsilon_{n-1} \quad (3.88) \]

The mean of \( X_n \) for the ARMA(1,1) process can be found using [17, p. 152] and is given by:

\[ E[X] = \frac{\mu}{1 - \phi_1} \quad (3.89) \]

The variance of the ARMA(1,1) process [17, p. 206] is given by:

\[ Var[X] = \frac{1 + \theta_1^2 - 2\theta_1 \phi_1}{1 - \phi_1^2} \sigma^2_{\varepsilon_n} \quad (3.90) \]

3.11.1.2 Autoregressive (AR) Process

In the case where \( \Theta(B) = 1 \) the process is a pure autoregressive AR(p) process given by:

\[ \Phi(B)(X_n - \mu) = \varepsilon_n \quad (3.91) \]

The special case where \( p = 1 \) (that is \( \Phi(B) = 1 - \phi_1 B^1 \)) and \( X_t \) has zero mean gives rise to the AR(1) process, which is a Markov process:

\[ X_n = \phi_1 X_{n-1} + \varepsilon_n \quad (3.92) \]

For this process the current value of the time-series \( X_n \) only depends on a fraction of the previous value plus a random shock \( \varepsilon_n \).

For a pure autoregressive AR(p) model, the PACF cuts off abruptly at lags greater than \( p \) whilst the ACVF decays exponentially [17, p. 214].

3.11.1.3 Moving Average (MA) Process

In the case where \( \Phi(B) = 1 \), the process is a pure moving average MA(p) process. In this case, the model is given by:

\[ (X_n - \mu) = \Theta(B)\varepsilon_n \quad (3.93) \]

When the process has zero mean then 3.93 becomes:
3 Preliminaries and Methodology

\[ X_n = \Theta(B)\varepsilon_n \quad (3.94) \]

The ACF of an MA(q) process cuts off abruptly after lag q [17, p. 214].

3.11.2 Autoregressive Integrated Moving Average Process

ARMA models are used when the time-series is stationary. Where the time-series has a drift (trend), differencing can be employed in order to obtain a stationary series. Differencing may need to be applied several times in order to achieve stationarity. When differencing is employed, the process is modelled by an Autoregressive Integrated Moving Average ARIMA(p,d,q) process.

The GARMA process becomes an ARIMA when \( \eta = 1 \), \( \delta = \frac{1}{2} \) in equation 3.80 [112] and it is defined as:

\[ \Phi(B)\nabla^d (X_n - \mu) = \Theta(B)\varepsilon_n \quad (3.95) \]

In this case, \( \nabla^d = (1 - B)^d \) is the differencing operator and \( d \) is restricted to integer values. Here, the differencing parameter \( d \) is the number of times that the time-series is differenced, for example if \( d = 1 \), the differences between consecutive observations are taken while for \( d = 2 \), the differences of the differences are used and so on.

3.11.3 Long-Range Dependent (LRD) Time-series Models

While SRD processes are characterised by ACVFs that die off exponentially, LRD processes, in contrast, have ACVFs that display considerable persistence and tend to die off in a hyperbolic fashion. For time-series that exhibit LRD a FARIMA(p,d,q) or ARFIMA(p,d,q) model ([62, 74]) is more appropriate.

In general, for a time-series \( X_n \) that only exhibits LRD, the periodicity parameter \( \eta = 0 \) thus the GARMA model becomes:

\[ \Phi(B)\nabla^d (X_n - \mu) = \Theta(B)\varepsilon_n \quad (3.96) \]

This is similar to the expression of an ARIMA process (equation 3.95), however, in this case the differencing parameter \( d \) is not restricted to only integer values, but can take fractional values. This process is called a FARIMA(p,d,q) process. For this case, the differencing operator is given by:
\[ \nabla^d = (1 - B)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k \quad 0 < d \leq \frac{1}{2} \] (3.97)

The FARIMA(p,d,q) model is flexible since adjusting its parameters enables the model to be suitable for a range of time-series irrespective of whether they exhibit LRD or not.

### 3.12 Collection of Traffic Traces

The traffic traces used in this study have been obtained from the ‘SONG’ database that is part of the Game Environments Internet Utilisation Study (GENIUS) project [27]. The database has a collection of traffic traces for seven popular FPS games: Half-Life Counter-Strike (HLCS), Counter-Strike-Source (HL2CS), Half-Life (HLD), Half-Life 2 (HL2DM), ETPro, Quake3, and Quake 4 (Quake4). These traffic traces were collected between January and July 2006. All the traffic traces were obtained by passive measurements from games that were run under controlled conditions with the number of players participating ranging from two to nine players. The configuration of the game and that of the server and client machines for each of the game trials is described in the respective technical report for the particular game trial, details can be found at the Simulating Online Networked Games Database [28].

The HLDM, HL2DM, Quake3, and Quake4 game traces were obtained in ‘death-match’ mode. In this mode, all of the players are fighting against each other. HLCS, ETPro, and HL2CS traces were gathered in ‘team-play’ mode. In this mode, the players form two teams and fight against each other. The duration of each game trial was between fifteen and twenty minutes. Again, precise details are available in the technical reports.

All of the game clients were connected to a central server via a 100 Mbps Fast Ethernet switched LAN. The traffic was captured by using a PC configured as a bridge for ETPro, HLDM, HL2DM, HL2CS and Quake4, refer to Figure 3.3, or the traffic was captured directly by the game server for Quake3 and HLCS.

The traffic capture PC was based around an Intel Pentium III 800MHz (8kB L1 cache, 256kB L2 cache) with 128MB PC133 SD-RAM. Tcpdump [77] was used to collect a raw packet trace of the traffic on the network during each trial. Tcpdump was configured to capture the first 100 bytes of each frame, which consisted of
### Table 3.1: Time Stamp ing Accuracy [119]

<table>
<thead>
<tr>
<th>Actual IAT (ms)</th>
<th>Size (Bytes)</th>
<th>60</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>1.006</td>
<td>1.009</td>
<td>1.017</td>
<td>1.025</td>
<td>1.033</td>
</tr>
<tr>
<td></td>
<td>99th Percentile</td>
<td>1.009</td>
<td>1.016</td>
<td>1.024</td>
<td>1.032</td>
<td>1.040</td>
</tr>
<tr>
<td>50.0</td>
<td>Mean</td>
<td>50.006</td>
<td>50.009</td>
<td>50.017</td>
<td>50.025</td>
<td>50.033</td>
</tr>
<tr>
<td></td>
<td>99th Percentile</td>
<td>50.013</td>
<td>50.016</td>
<td>50.024</td>
<td>50.032</td>
<td>50.041</td>
</tr>
</tbody>
</table>

Figure 3.3: Traffic Capture Set-up

The raw traffic traces were filtered to remove frames that were not associated with either the clients or the game server. Finally, the traces were split according to the direction of the individual packet flows, i.e. server-to-client and client-to-server flows.

In general, the methodology (based on [16]) outlined below has been followed where a probability distribution model is fitted to the empirical data. However, the ACFs or spectra will also be used to check the time-series behaviour of the traffic. There will also be cases where a specific methodology needs to be employed for a particular case. The details of these methods are presented when necessary. Many of the functions to fit distributions, calculate the MLE etc. are available in either the MATLAB [96] or R [111] software packages, thus most of the modelling has been performed using either MATLAB or R.
The traffic generated between the game server and the clients is collected with a test bed similar to that shown in Figure 3.3 (empirical traces). Specific details are available from [28].

Using the empirical traffic traces:

- A standard distribution is chosen based on the examination of the traffic’s empirical PDF or CDF.
- Where the empirical distribution has ‘spikes’, these are modelled by degenerate distributions (deterministic) see Section 3.3.1.
- The chosen distribution’s parameters are estimated by using a MLE (refer to Section 3.5.2).
- The fit of the distribution is checked by either comparing the analytical and empirical PDFs or CDFs, or by plotting a Q-Q plot (Section 3.6). Where there is a significant deviation between the empirical and analytical distributions, Borella [16] suggests that the data set should split into parts and each part be modelled by separate distributions.
- A goodness-of-fit metric should be used to determine how well the model agrees with the empirical data. In this case, the \( \lambda^2 \) measure (Section 3.7) is used.

### 3.13 Summary

This chapter has presented a review of the basic statistical techniques and distribution functions that will be used throughout this work. In addition, a brief description of the data collection process and a brief outline of the methodology that will be followed in the remainder of this work have been outlined.

Important results from this section are the formulation of various novel mixture distributions: the \( \Gamma MG \) and the \( VG \). These distributions have properties that are useful for the modelling of the traffic in the subsequent chapters.

The next chapter deals with the analysis of the traffic traces for the various FPS games that form part of this work.
Chapter 4

FPS Games Traffic Traces

4.1 Introduction

This chapter gives an overview of the traffic generated by seven FPS games during the game-play phase, the descriptive statistics of the traffic traces, and a description the traffic parameters.

The rest of the chapter is organized as follows. In Section 4.2, a brief review the FPS game traffic is presented. Section 4.3 presents a brief review of the inter-arrival time for the seven games. In Section 4.4, the descriptive statistics for the various game traces are presented and discussed. Section 4.5 presents a brief discussion of the results, and summary is presented in Section 4.6.

4.2 First-person Shooter Traffic

Typical FPS game servers send game state information to each client so that they can update their own version of the game’s state with the actions taken by the other players. Typically, a server sends these updates as ‘back to back’ unicast packets at regular intervals that typically don’t dependent on the level of activity in the game [87, 88]. The time between these update packets is often configurable, thus the inter-arrival times may be deterministic. The clients also send update packets back to the server that contain information about the actions of their human players. These client-to-server updates are sent at intervals that are often influenced by the client’s hardware and the amount of player activity [87, 88]. As every client has a different perspective of the game’s virtual world, the server customises an update packet for the intended client. Because of human factors, the players’ behaviour
cannot be predicted with certainty, and because of the compression process (Section 2.3.2), the sizes of the update packet’s payload produced by both the client and the server are typically non-deterministic.

In the above context, the parameters needed for modelling the traffic flowing during the game-play phase are the inter-arrival times and the payload size of the update packets that flow between the server and client. Several authors have presented models for these parameters (see Section 2.7); however, much of the work so far has been based on traffic traces that have been collected from Internet hosted game servers, thus the game trials were not usually run under controlled conditions or in a controlled environment. In some cases, the traces were collected under controlled conditions, but only for a few experiments [16] or with limited numbers of players. The traces used for this study were obtained from game trials that were conducted under strictly controlled and well documented conditions for varying numbers of players ranging from two to nine [28]. FPS games can typically involve up to 32 players, however, the trials in this study were limited to a maximum of nine players. In the next sections, the traffic traces for the various games will be analysed so that the identification of the key statistical features can be determined.

4.2.1 Traffic Traces

The descriptive statistics for the empirical traffic traces for the seven FPS games: Half-Life (HLDM), Half-Life Counter-Strike (HLCS), Quake III Arena (Quake3), Quake IV (Quake4), Wolfenstein Enemy Territory Pro (ETPro), Half-Life 2 (HL2DM) and Counter-Strike-Source (HL2CS) are presented in Sections 4.4.1 to 4.4.7. The traces were collected as outlined in Section 3.12.

These traces are examined in order to investigate which of the traffic parameters are important for the development of a suitable FPS traffic models. As modelling the random nature of the game traffic is of primary interest, any trends or periodicities in the traffic traces must be identified and removed when modelling the stochastic behaviour of the traffic. If these models are to form the basis of simulation models, then the deterministic and periodic characteristics of the game’s traffic can be re-incorporated into the simulation models as required.

4.2.2 Truncating the Traffic Traces

It was shown in Section 2.4 that the FPS games have three distinct phases. The start of a game consists of downloading data pertaining to the game, this has been
well modelled elsewhere. The end of the game generally consists of keep-alive information. The active game-play phase is usually the longest and most important phase of the game. In the start-up and end-game phases, there is little variation in the packet size, as can be seen in Figure 4.1 (a). As the game-play phase is the phase that affects the player’s experience of the game, this work will only model the traffic during this phase. Figure 4.1 (a) shows the first and last 2000 packets from the server-to-client trace of a two-player Wolfenstein Enemy Territory Pro game. Both the start-up and end-game phases are clearly evident. The other games have similar features with Half-Life 2 having the longest start-up phase of 1600 samples. In order to retain only the game-play phase traffic data, the samples corresponding to the start-up and end-game phases will be removed from the traffic trace. The traffic traces truncated by deleting 2000 samples from either side of the trace to retain only the game-play phase (Figure 4.1 (b)).

The next sections present the descriptive statistics of the traffic parameters for the seven FPS games.

4.3 Packet Inter-arrival Times

The packet Inter-arrival times (IATs) (or inter-packet intervals) of various FPS games have been studied by several authors, (refer to sections 2.7.3 to 2.7.5 for
a more complete discussion). Most authors have found that the inter-arrival times of the packets in the server-to-client direction are, in general, deterministic and show little variation, thus they are well modelled by either degenerate or discrete distributions. In the client-to-server direction, the inter-arrival times show some variation. For some FPS games, the update period is configurable, therefore any randomness in the inter-arrival times is due to either the delays introduced by the platform’s hardware or the various delays introduced in the network (serialisation, other traffic, etc.) [87]. Table 4.1 shows the typical inter-arrival times with an approximate value of the mode and/or distribution for the various games that form part of this study. These inter-arrival times have been measured at the server end. The distribution of the client-to-server inter-arrival times for Counter-Strike-Source and Half-Life 2 is modelled well by an Extreme Value distribution, as the variance is quite large. For Wolfenstein Enemy Territory Pro, the distribution of the client-to-server inter-arrival times is modelled by an Extreme Value and a degenerate distribution. Some games periodically exchange small packets between the server and client which results in a multi-modal distribution (for example Quake4).

An interesting observation from the data is that for certain games the inter-arrival times are bimodal, according to Armitage et al. [8, p. 155] this is due to the server mixing the transmission rates in order to achieve the client’s requested snapshot rate. This effect can be modelled in a simulator by specifying the client snapshot rate, and the server tick time using the following relationships:

$$
\frac{1}{\alpha + 1} \ STT \leq \ CSR \leq \frac{1}{\alpha \ STT}
$$  \hspace{1cm} (4.1)

$$
\alpha = \text{floor} \left( \frac{1}{STT \times CSR} \right)
$$

$$
X = \alpha - \left( \frac{1}{STT \times CSR} \right) + 1
$$  \hspace{1cm} (4.2)

Where $CSR$ is the client snapshot rate, $STT$ is the server tick time, $\alpha$ is an integer and $X$ is the proportion of packets sent with an inter-arrival time of $\alpha \ STT$.

The packet inter-arrival times form an integral part of the traffic models. However, as the inter-arrival times are essentially deterministic, they will not be analysed further in detail.
4.4 Payload Size

The update packets flowing between the server and the client show considerable variation in their UDP payload size (refer to Sections 2.7.4 and 2.7.6). This traffic parameter is of great significance as it is non-deterministic, for reasons that have been discussed previously. Since we are interested in a parameter that is indicative of the level of activity in the game, we will use the UDP payload size of the IP packets in our analysis. The layer two and three headers have been removed as these have a constant length. If required the length of the IP and the appropriate layer 2 headers can be added for simulation purposes.

For simplicity, the PMF or the PDF will be referred to as the density in the following discussion, irrespective of whether the distribution is discrete or continuous and the CDF will be referred to as the distribution. The un-sanitised trace will be referred to the ‘raw’ trace and the sanitised trace as the ‘clean’ trace in the following discussions.

4.4.1 Descriptive Statistics - Quake III Arena

‘Quake III Arena’ [75] is an FPS game that was released in late 1999 and developed by ‘id Software’. This game is built on the ‘Id Tech 3’ game engine. Quake III Arena is a team-oriented on-line FPS game that involves different game-play modes: in ‘Free for All’ each player competes against all other players; in ‘Team Death-match’ two teams compete against each other; in ‘Tournament’ two players compete against each other for a set time; and in ‘Capture the Flag’ two teams compete with each other to achieve a set mission objective. The game can support up to 32 players in a single game. The traces for Quake III Arena (‘caialab3’ map) were collected on the 10th and 11th January 2006 [124].

Table 4.1: Inter-arrival time

<table>
<thead>
<tr>
<th>Game</th>
<th>mode S2C IAT (ms)</th>
<th>mode C2S IAT (ms)</th>
<th>S2C / C2S Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake3</td>
<td>50</td>
<td>10.74</td>
<td>degenerate / degenerate</td>
</tr>
<tr>
<td>HLCS</td>
<td>50, 60</td>
<td>33, 50</td>
<td>discrete / discrete</td>
</tr>
<tr>
<td>Half-Life</td>
<td>50, 60</td>
<td>33, 50</td>
<td>discrete / discrete</td>
</tr>
<tr>
<td>HL2CS</td>
<td>30, 60</td>
<td>-</td>
<td>discrete / extreme(42.3,34.7)</td>
</tr>
<tr>
<td>HL2DM</td>
<td>30, 45</td>
<td>-</td>
<td>discrete / extreme(40.7,16.2)</td>
</tr>
<tr>
<td>ETPro</td>
<td>50</td>
<td>10.74</td>
<td>degenerate / Extreme(14.98,2.75) + degenerate</td>
</tr>
<tr>
<td>Quake4</td>
<td>80 (15, 33, 47)</td>
<td></td>
<td>discrete / degenerate (1.7) + gaussian(21.4,3.6)</td>
</tr>
</tbody>
</table>

4.4 Payload Size
4.4.1.1 Quake III Arena: Server-to-Client Payload Size

The descriptive statistics for the Quake III Arena server-to-client payload size are shown in Table 4.2. The trace was truncated to remove the start-up and end-game phases, as per Section 4.2.2. Truncating the trace reduces the variability of the payload size (Coefficient of variation (CV)), as a large number of small packets (<19 bytes) and a number of large packets are discarded. (Refer to the density plots of Figure 4.3.) The mean, median, and variance of the server-to-client packet payload are plotted in Figure 4.2. From the results, it can be seen that Quake III Arena is characterised by both a mean and median packet payload size that increases linearly with the number of players. The variance of the payload size also increases linearly with the number of players. This relationship is interesting, as it raises the question of whether the payload size distributions for the different numbers of players can be derived from games with a small number of players. This is important in the context of deriving a traffic model from a small number of measurements. We will explore this issue in considerable depth in Chapter 7.

Several authors have stated that the server-to-client traffic for FPS games is correlated [16, 19]. A ‘runs test’ [82, p. 123] was used to check whether the payload sizes were independent for the server-to-client packet traces of all of the seven FPS games. The ‘runs test’ reported that the samples were not independent at the 95% confidence level, confirming the results found by other authors. The ACVF can also be used to determine whether the payload sizes are independent [19]. The ACVF (Section 3.8) will be used as it provides insights into the nature of the trace. Figure 4.5 shows the ACVF for a 2-player and a 9-player game. Superimposed on the ACVF is a best-fit exponential. It can be seen that the ACVF has a high value for small lags and dies off faster than the exponential for lags up to at least 50 packets. Examination of the spectrum (Figure 4.4) does not reveal any significant periodicities for the 2-player game. Given this observation, it seems plausible that the server-to-client packet payload exhibits short range dependent effects. However, the ACVF of the 9-player game reveals a ‘hump’ at a lag of approximately 250 packets and there seems to be a high frequency periodicity in the trace for this game. These effects are not large, so they will not be considered further.
4.4 Payload Size

Table 4.2: Descriptive Statistics Quake III Arena: S2C Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Payload Size (Bytes)</th>
<th>Raw (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Players</td>
<td>Range</td>
</tr>
<tr>
<td>2</td>
<td>2473</td>
<td>24373</td>
<td>15-1308</td>
</tr>
<tr>
<td>3</td>
<td>24225</td>
<td>24225</td>
<td>15-1308</td>
</tr>
<tr>
<td>4</td>
<td>24116</td>
<td>24116</td>
<td>15-1308</td>
</tr>
<tr>
<td>5</td>
<td>24140</td>
<td>24140</td>
<td>15-1308</td>
</tr>
<tr>
<td>6</td>
<td>24149</td>
<td>24149</td>
<td>15-1308</td>
</tr>
<tr>
<td>7</td>
<td>24124</td>
<td>24124</td>
<td>15-1308</td>
</tr>
<tr>
<td>8</td>
<td>24100</td>
<td>24100</td>
<td>15-1308</td>
</tr>
<tr>
<td>9</td>
<td>23955</td>
<td>23955</td>
<td>15-1308</td>
</tr>
</tbody>
</table>

Mean Pkt size: $y=19.29n+18.68$  
Median Pkt size: $y=20.35n+21.38$

Variance of Pkt size: $y=539.0n+482.8$

Figure 4.2: Quake III Arena: S2C Mean, Median and Variance vs Number of Players

Figure 4.3: Quake III Arena: Density of S2C Payload Size
4.4.1.2 Quake III Arena: Client-to-Server Payload Size

Descriptive statistics for the Quake III Arena client-to-server payload size of the raw trace are given in Table 4.3. In this case, truncating the trace to ensure that only the game-play phase is included results in a significant reduction in the range of the packet’s payload sizes and the CV. There is little or no variation neither in the mean and median payload size, nor in the variance for the differing number of players. The work of Branch and Armitage [20] supports this observation, their explanation is that the behaviour of the individual players does not change as the number of players changes, thus the size of this packet does not depend on the number of players. In
4.4 Payload Size

Table 4.3: Descriptive Statistics Quake III Arena C2S: Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Range (Bytes)</th>
<th>Raw (Bytes)</th>
<th>Truncated (Bytes)</th>
<th>Range</th>
<th>Mean</th>
<th>Median</th>
<th>Variance</th>
<th>CV</th>
<th>Range</th>
<th>Mean</th>
<th>Median</th>
<th>Variance</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>110398</td>
<td>11-461</td>
<td>34.0</td>
<td>35</td>
<td>11.3</td>
<td>0.10</td>
<td>25-45</td>
<td>34.2</td>
<td>35</td>
<td>8.4</td>
<td>0.08</td>
<td>5</td>
<td>25-45</td>
<td>8.1</td>
</tr>
<tr>
<td>3</td>
<td>109498</td>
<td>11-429</td>
<td>32.7</td>
<td>33</td>
<td>13.3</td>
<td>0.11</td>
<td>25-43</td>
<td>32.8</td>
<td>33</td>
<td>8.1</td>
<td>0.09</td>
<td>5</td>
<td>25-43</td>
<td>8.1</td>
</tr>
<tr>
<td>4</td>
<td>108780</td>
<td>11-431</td>
<td>33.5</td>
<td>34</td>
<td>14.3</td>
<td>0.11</td>
<td>25-45</td>
<td>33.6</td>
<td>34</td>
<td>7.6</td>
<td>0.08</td>
<td>5</td>
<td>25-45</td>
<td>7.6</td>
</tr>
<tr>
<td>5</td>
<td>107100</td>
<td>11-433</td>
<td>33.2</td>
<td>33</td>
<td>16.6</td>
<td>0.12</td>
<td>25-43</td>
<td>33.2</td>
<td>33</td>
<td>8.1</td>
<td>0.09</td>
<td>5</td>
<td>25-43</td>
<td>8.1</td>
</tr>
<tr>
<td>6</td>
<td>105486</td>
<td>11-439</td>
<td>33.7</td>
<td>34</td>
<td>15.1</td>
<td>0.12</td>
<td>25-43</td>
<td>33.8</td>
<td>34</td>
<td>7.9</td>
<td>0.08</td>
<td>5</td>
<td>25-43</td>
<td>7.9</td>
</tr>
<tr>
<td>7</td>
<td>104131</td>
<td>11-452</td>
<td>33.4</td>
<td>34</td>
<td>10.4</td>
<td>0.10</td>
<td>25-43</td>
<td>33.5</td>
<td>34</td>
<td>7.8</td>
<td>0.08</td>
<td>5</td>
<td>25-43</td>
<td>7.8</td>
</tr>
<tr>
<td>8</td>
<td>105055</td>
<td>11-434</td>
<td>32.6</td>
<td>32</td>
<td>13.9</td>
<td>0.11</td>
<td>25-43</td>
<td>32.6</td>
<td>32</td>
<td>8.3</td>
<td>0.09</td>
<td>5</td>
<td>25-43</td>
<td>8.3</td>
</tr>
<tr>
<td>9</td>
<td>104578</td>
<td>11-437</td>
<td>32.6</td>
<td>32</td>
<td>11.3</td>
<td>0.10</td>
<td>25-45</td>
<td>32.7</td>
<td>32</td>
<td>8.7</td>
<td>0.09</td>
<td>5</td>
<td>25-45</td>
<td>8.7</td>
</tr>
</tbody>
</table>

Figure 4.6: Quake III Arena: ACVF C2S Payload Size: Truncated Trace

addition, this observation is reasonable since the client update packets are made up of the current changes in the view of the particular player, hence they are not dependent on the total number of players participating in the game. Figure 4.6 shows the ACVF for a 2-player and a 7-player game with the best-fit exponential, it can be seen that the ACVF starts at a low value and is approximately 1 after 100 packets.

4.4.2 Descriptive Statistics - Wolfenstein Enemy Territory Pro

‘Wolfenstein Enemy Territory Pro’ [116] is a modification (‘mod’) of Wolfenstein Enemy Territory that has been designed for competition play between several players. This game was developed by ‘Splash Damage Ltd’ and released in 2003. Wolfenstein
Enemy Territory Pro is a team-oriented on-line FPS game that involves two teams (Allies and Axis). Wolfenstein Enemy Territory Pro can support up to 32 players in a single game. This game is based on a modified ‘Return To Castle Wolfenstein’ game engine, which is based on the ‘Id Tech 3’ engine. The traces for Wolfenstein Enemy Territory Pro were collected on 18th and 19th January 2006 and are based on the ‘fueldump’ map[120].

4.4.2.1 Wolfenstein Enemy Territory Pro: Server-to-Client Payload Size

Descriptive statistics for the Wolfenstein Enemy Territory Pro server-to-client payload size of the raw and truncated traces are shown in Table 4.4. The trace has been truncated to remove the start-up and end-game phases, as was the case for Quake III Arena. From Figure 4.7, we see that this game is also characterised by both a mean and median payload size that increases linearly with the number of players. The variance also increases linearly with the number of players.

The density of the server-to-client payload size of the 2 player game raw trace (Figure 4.8) shows that the server frequently sends very small packets of 18 or 19 bytes, and a significant number of larger packets of 236, 473, and 1306 bytes. Truncating the series by removing the start-up and end-game phase has the effect of removing the ‘spikes’ in the density that are due to the very small and large packets (Table 4.4) that are exchanged during these phases of the game.

The ACVF plot (Figure 4.10) of the ETPro server-to-client trace dies off exponentially for both the 2-player and 9-player case. There is a deviation from the exponential for the 9-player game after a lag of 300 packets. Both the ACVF (Figure 4.10) and the spectrum plot (Figure 4.9) show that there is a significant periodic component (every 41 packets) in the trace; this periodicity needs to be accounted for in the model for the traffic of this game. There are also some other periodic components of lower amplitude. Bussiere and Zander [25] noted that for this game the server sends 50 byte packets to the client when it is inactive. Further investigation using classical decomposition techniques [24, p. 14] show that the periodic component is due to an addition of 50 bytes to the packet’s payload that is sent every 41 packets and not just during the periods of inactivity. The trace can be de-seasonalised by subtracting 50 bytes from every integer multiple of 41 packets. If need be, this periodic component can be re-introduced into the trace by adding it back into the trace. De-seasonalising the trace removes the periodic component as
### 4.4 Payload Size

#### Table 4.4: Descriptive Statistics Wolfenstein Enemy Territory Pro: S2C Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Raw (Bytes)</th>
<th>Truncated (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Range</td>
<td>Mean</td>
</tr>
<tr>
<td>2</td>
<td>24660</td>
<td>18-1308</td>
<td>74.8</td>
</tr>
<tr>
<td>3</td>
<td>24498</td>
<td>18-1308</td>
<td>101.6</td>
</tr>
<tr>
<td>4</td>
<td>23774</td>
<td>18-1308</td>
<td>115.7</td>
</tr>
<tr>
<td>5</td>
<td>24638</td>
<td>18-1308</td>
<td>140.3</td>
</tr>
<tr>
<td>6</td>
<td>23724</td>
<td>17-1308</td>
<td>178.2</td>
</tr>
<tr>
<td>7</td>
<td>24616</td>
<td>18-1308</td>
<td>181.7</td>
</tr>
<tr>
<td>8</td>
<td>24731</td>
<td>18-1308</td>
<td>192.4</td>
</tr>
<tr>
<td>9</td>
<td>24747</td>
<td>18-1308</td>
<td>234.9</td>
</tr>
</tbody>
</table>

![Graph showing Mean and Median of S2C Payload Size](image1)

![Graph showing Variance of S2C Payload Size](image2)

Figure 4.7: Wolfenstein Enemy Territory Pro: S2C Mean, Median and Variance vs Number of Players

...can be seen in the ‘clean’ spectrum and ACVF of Figure 4.11. Once the data has been de-seasonalised, the match between the ACVF and the exponential is excellent.
Figure 4.8: Wolfenstein Enemy Territory Pro: Density of S2C Payload Size

Figure 4.9: Wolfenstein Enemy Territory Pro: Spectrum S2C Raw Payload Size

Figure 4.10: Wolfenstein Enemy Territory Pro: S2C Payload Size: Truncated Raw Trace
4.4 Payload Size

Figure 4.11: Wolfenstein Enemy Territory Pro: Density, Spectrum S2C Payload Size: De-seasonalised Trace

![Graph showing PDF of ETPro clean and Power Spectrum of S2C Payload ETPro 2 players clean.]

Figure 4.12: Wolfenstein Enemy Territory Pro: ACVF S2C Payload Size: De-seasonalised Trace

4.4.2.2 Wolfenstein Enemy Territory Pro: Client-to-Server Payload Size

The descriptive statistics for the Wolfenstein Enemy Territory Pro client-to-server payload size of the raw and truncated traces are given in Table 4.5. In this case, truncating the trace to ensure that only the game-play phase is included results in a significant reduction in the range of the packet’s payload sizes. As was the case with Quake III Arena, again there is little or no variation in either the mean, median or variance of the payload size. Figure 4.13 shows the ACVF for a 2-player and a 9-player game with the best-fit exponential, we see that the ACVF starts at a low value and dies off slowly, but is not significant after approximately 100 packets. This low autocovariance indicates that there is a weak relationship between the current
Table 4.5: Descriptive Statistics Wolfenstein Enemy Territory Pro: C2S Trace

| Players | Samples (Raw) | Raw (Bytes) | | | | Truncated (Bytes) | | | |
|---------|---------------|-------------|-------------|-------------|-------------|---------------|-------------|-------------|-------------|-------------|
|         | Range         | Mean        | Median      | Variance    | CV          | Range         | Mean        | Median      | Variance    | CV          |
| 2       | 89179         | 11 - 290    | 32.3        | 32          | 14.2        | 27 - 47       | 32.5        | 32          | 7.7         | 0.09        |
| 3       | 93026         | 11 - 294    | 32.6        | 32          | 14.7        | 27 - 53       | 32.6        | 32          | 7.6         | 0.08        |
| 4       | 70922         | 11 - 298    | 32.5        | 32          | 15.9        | 27 - 43       | 32.6        | 32          | 8.6         | 0.09        |
| 5       | 80223         | 11 - 293    | 32.8        | 32          | 20.3        | 27 - 47       | 32.9        | 32          | 9.0         | 0.09        |
| 6       | 64254         | 11 - 298    | 33.2        | 33          | 19.6        | 27 - 47       | 33.3        | 33          | 9.0         | 0.09        |
| 7       | 68999         | 11 - 293    | 33.4        | 33          | 26.2        | 27 - 47       | 33.4        | 33          | 9.5         | 0.09        |
| 8       | 69091         | 11 - 290    | 33.4        | 33          | 21.0        | 26 - 49       | 33.5        | 33          | 10.3        | 0.10        |
| 9       | 58913         | 11 - 291    | 33.4        | 32          | 29.5        | 27 - 47       | 33.5        | 33          | 10.1        | 0.09        |

Figure 4.13: Wolfenstein Enemy Territory Pro: ACVF C2S Payload Size: Truncated Trace

packet size and the previous ones. The ACVF also reveals some small periodic effects in the traces.

4.4.3 Descriptive Statistics Half-Life 2

‘Half-Life 2’ [133] is an FPS game that was developed by Valve Software and released in 2004. This game is based on the ‘Source’ game engine. Half-Life 2 can be played in either free-for-all or team-based modes. The traces for Half-Life 2 were collected between the 3rd and 16th of February 2006 and are based on the ‘overwatch’ map[122].
4.4 Payload Size

Table 4.6: Descriptive Statistics Half-Life 2: S2C Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Raw (Bytes)</th>
<th>Truncated (Bytes)</th>
<th>Range</th>
<th>Mean</th>
<th>Median</th>
<th>Variance</th>
<th>CV</th>
<th>Range</th>
<th>Mean</th>
<th>Median</th>
<th>Variance</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>36402</td>
<td>16-1400</td>
<td>95.9</td>
<td>86</td>
<td>3289.8</td>
<td>95.9</td>
<td>39-1239</td>
<td>3996.3</td>
<td>0.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>36473</td>
<td>16-1400</td>
<td>122.1</td>
<td>109</td>
<td>3095.6</td>
<td>123.5</td>
<td>39-1400</td>
<td>3992.2</td>
<td>0.51</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>36819</td>
<td>16-1307</td>
<td>154.9</td>
<td>142</td>
<td>6505.8</td>
<td>150.7</td>
<td>39-1000</td>
<td>4997.2</td>
<td>0.44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>36821</td>
<td>16-1400</td>
<td>157.1</td>
<td>145</td>
<td>7782.7</td>
<td>161.5</td>
<td>40-1400</td>
<td>6187.9</td>
<td>0.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>36822</td>
<td>16-1400</td>
<td>202.9</td>
<td>190</td>
<td>10050.7</td>
<td>212.5</td>
<td>40-1400</td>
<td>8690.1</td>
<td>0.44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>37037</td>
<td>16-1400</td>
<td>210.6</td>
<td>203</td>
<td>11250.6</td>
<td>222.3</td>
<td>41-1400</td>
<td>8534.6</td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>36335</td>
<td>16-1400</td>
<td>241.8</td>
<td>229</td>
<td>15063.1</td>
<td>251.0</td>
<td>40-1400</td>
<td>7811.0</td>
<td>0.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>36625</td>
<td>16-1400</td>
<td>277.1</td>
<td>269</td>
<td>17888.2</td>
<td>285.3</td>
<td>45-1400</td>
<td>13728.9</td>
<td>0.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.14: Half-Life 2: S2C Mean and Median vs Number of Players

4.4.3.1 Half-Life 2: Server-to-Client Payload Size

The descriptive statistics for the Half-Life 2 server-to-client payload size are shown in Table 4.6. Removing the start-up and end-game phases results in a reduction of the range of the payload size. This reduction is due to the removal of a large number of small packets (<35 bytes), refer to the density plots in Figure 4.15. From these results, it can be seen that this game is also characterised by both a mean and median packet size that increases with the number of players. The variance also increases linearly with the number of players (Figure 4.14), this relationship is similar to ETPro and Quake3.

Whilst examination of the spectrum (Figure 4.16) does not reveal any significant periodicities for this game, there is a ‘hump’ in the ACVF at a lag of approximately 150. Figure 4.17 shows the ACVF for a 2-player and a 9-player game with the best-fit exponential. From this, we see that the ACVF dies off exponentially for lags up to at least 150 packets. Given this observation, it seems plausible that the server-to-client payload exhibits some short-range dependent effects.
4.4.3.2 Half-Life 2: Client-to-Server Payload Size

The descriptive statistics for the Half-Life 2 client-to-server payload size of the raw trace are given in Table 4.7. For this game, truncating the trace so that only the game-play phase is included results in a significant reduction in the range of the packet’s payload sizes and the Coefficient of variation, refer to Table 4.7. There is little or no variation in either in the mean and median payload size or in the variance for the differing number of players for this game.

The 2-player a 9-player game ACVFs (Figure 4.18) show that it drops very quickly and becomes negligible after a lag of 100 packets.
4.4 Payload Size

4.4.4 Descriptive Statistics - Counter-Strike-Source

‘Counter-Strike-Source’ [132] is a very popular team-oriented on-line FPS game that involves two teams (terrorists and counter-terrorists) in combat with each other.
This game was developed by Valve Software and released in 2004 and it uses the ‘Source’ game engine. The traces for ‘Counter-Strike-Source’ were collected between the 7th and 15th of February 2006 and are based on the ‘dedust’ map [121].

4.4.4.1 Counter-Strike-Source: Server-to-Client Payload Size

The descriptive statistics for the ‘Counter-Strike-Source’ server-to-client payload size are shown in Table 4.8. Examination of the density of the server-to-client payload size of the raw trace (Figure 4.20) reveals that the server frequently sends very small packets of less than 30 bytes. The server also sends a small number of larger packets with sizes varying from 200 to 1400 bytes. Removing the start-up and end-game phases has the effect of reducing the variation in size of the packets (Figure 4.20) as the small packets are removed. From the results of the truncated trace, it can be seen that this game is characterised by both a mean and median packet size that increases with the number of players, albeit with a small gradient. The variance also increases with the number of players (Figure 4.19).

Figure 4.22 shows the ACVF for a 2-player and a 9-player game with the best-fit exponential, it can be seen that the ACVF dies off exponentially for lags up to at least 150 packets. Given this observation, it is plausible that the server-to-client payload exhibits some short-range dependent effects. Examination of the spectrum (Figure 4.21) does not reveal any significant periodicities for this game; however, examination of the ACVF plot reveals a peak at a lag of approximately 150, implying that this trace does contain some periodic packets.
4.4 Payload Size

Figure 4.19: Counter-Strike-Source: S2C Mean, Median and Variance vs Number of Players

Figure 4.20: Counter-Strike-Source: Density of S2C Payload Size

Figure 4.21: Counter-Strike-Source: Spectrum S2C Payload Size
Descriptive statistics for the ‘Counter-Strike-Source’ client-to-server payload size of the raw trace are given in Table 4.9. For this game, truncating the trace results in a reduction in both the range of the packet’s payload sizes and the CV (Table 4.9). There is little or no variation neither in the mean and median payload size nor in the variance for the differing number of players. Figure 4.23 shows the ACVF for a 2-player and a 9-player game with the best-fit exponential, it can be seen that the ACVF of the 2-player game drops dramatically after lag 1 and is approximately zero after 150 packets. However, for the 9-player, whilst the ACVF drops quickly for low lags, it continues to persist and dies off at a slower rate than the exponential.
4.4 Payload Size

4.4.5 Descriptive Statistics - Half-Life

‘Half-Life’ [131] is a FPS game that was developed by Valve Software and released in 1998. Half-Life can be played in either a ‘free-for-all’ or ‘team-based’ mode and it can support 32 players in a single game. Half-Life is based on the ‘GoldSrc’ game engine (a modified Quake engine). The traces for Half-Life were collected between the 12th and 13th of January 2006 and are based on the “frenzy” map [122].

4.4.5.1 Half-Life: Server-to-Client Payload Size

The descriptive statistics for the Half-Life server-to-client payload size are shown in Table 4.10. In this case, truncating the trace results in the removal of several small packets (<27 bytes) and some large packet (>992 bytes). This game is also characterised by both a mean and median payload size that increases linearly with the number of players (Figure 4.24). The variance also increases linearly with the number of players (Figure 4.24) in a similar way to the previous games.

Examination of the spectrum (Figure 4.26) does not reveal any significant periodicities for this game.

Figure 4.27 shows the ACVF for a 2-player and a 9-player game. As can be seen, the ACVF dies off exponentially for lags up to at least 500 packets, corresponding to durations up to about 25 seconds.
Table 4.10: Descriptive Statistics Half-Life: S2C Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Raw (Bytes)</th>
<th>Truncated (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Range</td>
<td>Mean</td>
<td>Median</td>
</tr>
<tr>
<td>2</td>
<td>16-1042</td>
<td>64.8</td>
<td>56.0</td>
</tr>
<tr>
<td>3</td>
<td>6-1042</td>
<td>81.6</td>
<td>69.0</td>
</tr>
<tr>
<td>4</td>
<td>16-1042</td>
<td>93.9</td>
<td>80.0</td>
</tr>
<tr>
<td>5</td>
<td>16-1042</td>
<td>93.9</td>
<td>82.0</td>
</tr>
<tr>
<td>6</td>
<td>16-1042</td>
<td>105.8</td>
<td>94.0</td>
</tr>
<tr>
<td>7</td>
<td>16-1042</td>
<td>119.8</td>
<td>108.0</td>
</tr>
<tr>
<td>8</td>
<td>16-1042</td>
<td>129.6</td>
<td>117.0</td>
</tr>
<tr>
<td>9</td>
<td>16-1042</td>
<td>138.5</td>
<td>127.0</td>
</tr>
</tbody>
</table>

Figure 4.24: Half-Life: S2C Mean, Median and Variance vs Number of Players

Figure 4.25: Half-Life: Density of S2C Payload Size
4.4 Payload Size

Figure 4.26: Half-Life: Spectrum S2C Payload Size

Figure 4.27: Half-Life: ACFV S2C Payload Size: Truncated Trace

Table 4.11: Descriptive Statistics Half-Life: C2S Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Raw (Bytes)</th>
<th>Truncated (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Range</td>
<td>Mean</td>
</tr>
<tr>
<td>2</td>
<td>28869</td>
<td>16 - 221</td>
<td>48.6</td>
</tr>
<tr>
<td>3</td>
<td>28785</td>
<td>16 - 222</td>
<td>49.2</td>
</tr>
<tr>
<td>4</td>
<td>19145</td>
<td>16 - 220</td>
<td>49.2</td>
</tr>
<tr>
<td>5</td>
<td>28601</td>
<td>16 - 223</td>
<td>49.5</td>
</tr>
<tr>
<td>6</td>
<td>21522</td>
<td>16 - 223</td>
<td>49.6</td>
</tr>
<tr>
<td>7</td>
<td>21529</td>
<td>16 - 222</td>
<td>49.3</td>
</tr>
<tr>
<td>8</td>
<td>21615</td>
<td>16 - 222</td>
<td>49.8</td>
</tr>
<tr>
<td>9</td>
<td>21604</td>
<td>16 - 220</td>
<td>49.8</td>
</tr>
</tbody>
</table>
4.4.5.2  Half-Life: Client-to-Server Payload Size

The descriptive statistics for the Half-Life Counter-Strike client-to-server payload size of the raw and truncated traces are given in Table 4.11. For this game, truncating the trace also results in a significant reduction in the range of the packet’s payload sizes and the CV. For this case, there is little or no variation in either the mean and median payload size or in the variance for the differing number of players, this indicates that the packet’s payload is independent of the number of players in the game. Figure 4.28 shows the ACVF for a 2-player and a 9-player game with the best-fit exponential, it can be seen that the ACVF starts at a low value (<20) and drops dramatically after lag 1 and is negligible after 100 packets. The low ACVF indicates that there is a weak relationship between the current packet size and the previous ones.

4.4.6  Descriptive Statistics - Half-Life Counter-Strike

‘Half-Life Counter-Strike’ [130] is a FPS game derived from Half-Life. This game was developed by Valve Software and released in 2000. Half-Life Counter-Strike is a team-oriented on-line FPS that involves two teams (terrorists and counter-terrorists) combating each other. Half-Life Counter-Strike is also based on the ‘GoldSrc’ game engine. The traces for Half-Life Counter-Strike were collected between the 13th and 17th of January 2006 and are based on the ‘dedust’ map [123].
4.4 Payload Size

Table 4.12: Descriptive Statistics Half-Life Counter-Strike: S2C Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Raw (Bytes)</th>
<th>Truncated (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Range</td>
<td>Mean</td>
</tr>
<tr>
<td>2</td>
<td>22814</td>
<td>16-1121</td>
<td>53.9</td>
</tr>
<tr>
<td>3</td>
<td>23196</td>
<td>16-1122</td>
<td>67.3</td>
</tr>
<tr>
<td>4</td>
<td>22782</td>
<td>6-1118</td>
<td>69.4</td>
</tr>
<tr>
<td>5</td>
<td>17256</td>
<td>6-1135</td>
<td>75.2</td>
</tr>
<tr>
<td>6</td>
<td>17207</td>
<td>6-1132</td>
<td>67.7</td>
</tr>
<tr>
<td>7</td>
<td>17151</td>
<td>6-1118</td>
<td>78.6</td>
</tr>
<tr>
<td>8</td>
<td>17146</td>
<td>6-1118</td>
<td>96.4</td>
</tr>
<tr>
<td>9</td>
<td>16716</td>
<td>6-1170</td>
<td>93.0</td>
</tr>
</tbody>
</table>

Figure 4.29: Half-Life Counter-Strike: S2C Mean, Median and Variance vs Number of Players

4.4.6.1 Half-Life Counter-Strike: Server-to-Client Payload Size

The descriptive statistics for the Half-Life Counter-Strike server-to-client payload size are shown in Table 4.12. Truncating the trace reduces the variability of the payload size and the CV, as a large number of small packets (<19 bytes) and a number of large packets are discarded. Half-Life Counter-Strike is also characterised by both a mean and median packet size that increases linearly with the number of players (Figure 4.29). The variance also increases linearly with the number of players (Figure 4.29). This is similar to the results for the previous games.

Examination of the spectrum (Figure 4.31) does not reveal any major periodicities for this game; however, examination of the ACVF reveals a peak at a lag of approximately 95 for the 2-player game. This correlation is due to two large packets (>300 bytes) sent at 100 packet intervals approximately every 700 packets.
4.4.6.2 Half-Life Counter-Strike: Client-to-Server Payload Size

The descriptive statistics for the Half-Life Counter-Strike client-to-server payload size of the raw trace are given in Table 4.13. Truncating the trace so that only the game-play phase is retained results in a significant reduction in the range of the packet’s payload sizes and the CV. The mean, median, and variance of the payload size do not vary significantly as the number of players is varied. The ACVF plots for the 2-player and 9-player games (Figure 4.33) show that correlation drops significantly after lag 1 and then seems to die off at an exponential rate.
4.4 Payload Size

Table 4.13: Descriptive Statistics Half-Life Counter-Strike: C2S Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Range</th>
<th>Raw (Bytes)</th>
<th>Truncated (Bytes)</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Range</td>
<td>Mean</td>
<td>Median</td>
<td>Variance</td>
</tr>
<tr>
<td>2</td>
<td>26115</td>
<td>16-238</td>
<td>43.3</td>
<td>40</td>
<td>58.6</td>
</tr>
<tr>
<td>3</td>
<td>25799</td>
<td>16-238</td>
<td>43.9</td>
<td>42</td>
<td>61.8</td>
</tr>
<tr>
<td>4</td>
<td>25553</td>
<td>16-238</td>
<td>42.7</td>
<td>40</td>
<td>59.2</td>
</tr>
<tr>
<td>5</td>
<td>19572</td>
<td>16-238</td>
<td>44.4</td>
<td>40</td>
<td>65.5</td>
</tr>
<tr>
<td>6</td>
<td>19499</td>
<td>16-239</td>
<td>42.5</td>
<td>39</td>
<td>68.3</td>
</tr>
<tr>
<td>7</td>
<td>19683</td>
<td>16-237</td>
<td>49.8</td>
<td>37</td>
<td>66.8</td>
</tr>
<tr>
<td>8</td>
<td>19397</td>
<td>16-238</td>
<td>42.6</td>
<td>40</td>
<td>61.6</td>
</tr>
<tr>
<td>9</td>
<td>19369</td>
<td>16-238</td>
<td>43.6</td>
<td>41</td>
<td>64.2</td>
</tr>
</tbody>
</table>

Figure 4.32: Half-Life Counter-Strike: ACVF S2C Payload Size: Truncated Trace

Figure 4.33: Half-Life Counter-Strike: ACVF C2S Payload Size: Truncated Trace
Table 4.14: Descriptive Statistics Quake IV: S2C Trace

<table>
<thead>
<tr>
<th>Players</th>
<th>Samples (Raw)</th>
<th>Raw (Bytes)</th>
<th>Truncated (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Range</td>
<td>Mean</td>
<td>CV</td>
</tr>
<tr>
<td>2</td>
<td>18009</td>
<td>14-1310</td>
<td>81.4</td>
</tr>
<tr>
<td>3</td>
<td>17486</td>
<td>13-1310</td>
<td>118.5</td>
</tr>
<tr>
<td>4</td>
<td>17636</td>
<td>14-1310</td>
<td>138.8</td>
</tr>
<tr>
<td>5</td>
<td>17696</td>
<td>14-1310</td>
<td>163.2</td>
</tr>
<tr>
<td>6</td>
<td>17377</td>
<td>13-1310</td>
<td>198.9</td>
</tr>
<tr>
<td>7</td>
<td>17834</td>
<td>13-1310</td>
<td>233.1</td>
</tr>
</tbody>
</table>

4.4.7 Descriptive Statistics - Quake IV

‘Quake IV’ [76] is an FPS game developed by ‘Raven Software’ and released in 2005. Quake IV is also a team-oriented on-line First-person Shooter with similar game modes as those found in Quake III Arena. This game can support up to 16 players in a single game. Quake IV is based on the ‘id Tech 4’ engine. The traces for Quake IV were collected between the 20th and 31st of July 2006 and are based on the ‘q4dm1’ map [125]; it should be noted that, unlike the other games, there is a maximum of seven players for the Quake IV trials.

4.4.7.1 Quake IV Server-to-Client Payload Size

The descriptive statistics for the Quake IV server-to-client payload size are shown in Table 4.14. In this case, truncating the trace results in the removal of several small packets (<24 bytes) and some large packets (>835 bytes). Even after truncating the trace, there are still a large number of small packets of less than 30 bytes, as shown by the probability density function (Figure 4.35). The mean, median and variance of the payload size increase linearly with the number of players for this game as is the case for the other games, see Figure 4.34.

The spectral plots (Figure 4.36) strongly suggest that there are significant periodic effects for this trace. Close inspection of the 2-player trace reveals that there are a large number of packets of sizes between 24 and 28 bytes that are sent approximately every 7th or 8th packet with an inter-arrival time of less than 80ms. This periodic effect is very pronounced in the ACVF plot (Figure 4.38). Removal of these packets results in a ‘clean’ spectrum and ACF, (Figure 4.37 and Figure 4.39). These periodic packets could be modelled by a degenerate distribution.

The ACVF plot (Figure 4.39) of the Quake4 server-to-client trace of the ‘clean’ trace dies off exponentially for both the 2-player and 7-player case (Figure 4.39).
4.4 Payload Size

4.4.7.2 Quake IV Client-to-Server Payload Size

The descriptive statistics for the Quake IV client-to-server payload size of the raw trace are given in Table 4.15. In this case truncating the trace to ensure that only the game-play phase is included results in a reduction in the range of the packet’s payload sizes, due to the removal of packets whose size is greater than 100 bytes. As
Figure 4.36: Quake IV: Spectrum S2C Payload Size

Figure 4.37: Quake IV: Spectrum S2C Payload Size: De-seasonalised Trace

Figure 4.38: Quake IV: ACVF S2C Payload Size: Raw Trace
4.4 Payload Size

in every other case, there is little or no variation in either the mean, median, or the variance of the payload size for the differing number of players. Figure 4.40 shows the ACVF for a 2-player and a 7-player game with the best-fit exponential, again the periodic component is evident for this case.
Table 4.16: Characteristics of S2C Traffic Traces

<table>
<thead>
<tr>
<th>Game</th>
<th>Mean Payload vs number of players</th>
<th>Variance Payload vs number of players</th>
<th>Periodicity</th>
<th>ACVF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake3</td>
<td>Linear</td>
<td>Linear</td>
<td>Not Significant</td>
<td>Approx. Exponential</td>
</tr>
<tr>
<td>ETFPr</td>
<td>Linear</td>
<td>Linear</td>
<td>Strong</td>
<td>Approx. Exponential</td>
</tr>
<tr>
<td>HL2DM</td>
<td>Linear</td>
<td>Linear</td>
<td>Not Significant</td>
<td>Approx. Exponential</td>
</tr>
<tr>
<td>HL2CS</td>
<td>Linear with small gradient</td>
<td>Increases slightly</td>
<td>Slight</td>
<td>Approx. Exponential</td>
</tr>
<tr>
<td>HLDM</td>
<td>Linear</td>
<td>Linear</td>
<td>Not Significant</td>
<td>Approx. Exponential</td>
</tr>
<tr>
<td>HLCS</td>
<td>Linear</td>
<td>Linear</td>
<td>Slight</td>
<td>Approx. Exponential</td>
</tr>
<tr>
<td>Quake4</td>
<td>Linear</td>
<td>Linear</td>
<td>Strong</td>
<td>Approx. Exponential</td>
</tr>
</tbody>
</table>

4.5 Discussion

From the results of the previous sections, it can be seen that the server-to-client inter-arrival times are by and large deterministic, and the client-to-server IATs although they have some spread it is not significant. This result is similar to that found in previous work by other authors. Another interesting feature, which has generally been overlooked in the past, is that there may be significant periodic traffic flowing between clients and server. This periodic traffic may be problematic, as it is non-stationary. The effects of this must be dealt with appropriately when either analysing or simulating the traffic. These findings are summarised in Table 4.16.

The results also show that while the distributions of the inter-arrival times are either deterministic or have small random variations, the payload size has a greater variability and the density is typically skewed.

For the seven games, the mean payload length of the server-to-client update packets is proportional to the number of players participating in the game, confirming the findings of Färber [59] and others. On the other hand, the size of the client-to-server packet’s payload is not dependent on the number of players in the game. Typically, the ACVF dies off exponentially in the server-to-client direction, but some of the traces have strong periodic components, which can be modelled by degenerate distributions. In general, the client-to-server payloads have small ACVFs that indicate that there is a weak relationship between the payload size of the current packet size and the previous ones. In contrast, the server-to-client payloads have larger ACVFs indicating a strong relationship between the current packet size and the previous ones. These effects should be included in the models for this class of traffic.
4.6 Summary

In this chapter, the descriptive statistics of the seven FPS games have been presented in order to determine which of the game-play traffic characteristics are important for deriving an FPS traffic model.

The important results from this chapter are:

- The distributions of the server-to-client inter-arrival times are by and large deterministic.
- The distributions of the client-to-server IATs although they have some spread it is not significant.
- Some games have significant periodic traffic flowing between the client and server. This periodic traffic may be problematic as these non-stationary effects must be dealt with appropriately when either analysing or simulating the traffic.
- We have presented a simple novel model for the case of where the distributions of the inter-arrival times are bimodal. In contrast, the payload size has a greater variability and the density is typically skewed.
- For the seven games, the mean payload length of the server-to-client update packets is proportional to the number of players participating in the game. On the other hand, the size of the client-to-server packet’s payload is not dependent on the number of players in the game.
- The ACVF dies off exponentially in the server-to-client direction, but some of the traces have strong periodic components.
- The client-to-server payloads typically have small ACVFs indicating that there is a weak relationship between the payload size of the current packet size and the previous ones.
- The server-to-client payloads have larger ACVFs indicating a strong relationship between the current packet size and the previous ones.

In the next chapter, standard distributions are fitted to the payload size of the two player games in both the client-to-server and server-to-client directions, in order to identify statistical distributions that may be suitable as models for the payload lengths of the traffic.
Chapter 5

Standard Distribution Models

5.1 Introduction

In the previous chapter, the descriptive statistics for the seven FPS games were presented. The main findings from the analysis were that the distributions of the payload lengths showed more variability than did the inter-arrival times, which are typically deterministic in the server-to-client direction. Therefore, the server-to-client payload size is of more importance for modelling. It was also found that the mean payload length of the server-to-client update packets is proportional to the number of players participating in the game, confirming the findings of Färber [59] and others, whilst the client-to-server payload is independent of the number of players participating in a particular game. The server-to-client update packet’s payload length also exhibits significant autocorrelation that in many cases dies off exponentially. In some cases, periodicities were present in this traffic, these need to be dealt with in any FPS game traffic model.

In this chapter, several distributions are fitted to the payload size of a two-player game in both the client-to-server and server-to-client directions, as this parameter of the traffic, unlike the inter-arrival time, is usually non-deterministic. The purpose of this chapter is to identify statistical distributions, superior to those already proposed in the literature that may be suitable as models for the payload traffic. In this context, ‘superior’ means distributions that match as least as well, but which also describe the underlying physical process and which have mathematical properties that can be used to advantage when extrapolating the models.

The rest of the chapter is organized as follows. Section 5.2 presents a brief review the results of modelling FPS game traffic payloads. Section 5.3 presents the results
from fitting the various distributions to the payloads of the FPS games. In Section 5.4, conclusions are presented.

5.2 Standard Distribution Based Models for Payloads of FPS Games

Most of the work in modelling the game-phase traffic produced by FPS games is based on [16]'s methodology as outlined in Section 2.7.1, with much of this modelling concentrating on fitting standard distributions to the traffic parameters of interest. The results of the previous chapter showed that the distribution of the server-to-client payload length is typically positively skewed. This parameter has commonly been modelled by either Extreme Value distributions (Quake [16], Half-Life Counter-Strike [59] and the FPS game model for IEEE 802.16m [129, p. 120], or by Log-Normal distributions (Half-Life [87] and Quake III [88]). The Extreme Value and Log-Normal distributions capture the skewed nature of the empirical data. However, few authors have tried to justify why the underlying process would lead to payload lengths that are distributed in this way.

An important question to pose is whether there is a relationship between the distribution of the payload lengths of the packets flowing from the client to the server and those from the server back to the client. If so, and if this relationship is somehow an aggregation process, (which Feng et al. [53] suggest that it is), then would a combination of different distributions (a mixture distribution) be a more appropriate model? To answer this question, an appropriate mixture distribution must first be identified. In the next sections, various distributions are fitted to the payload size to determine if indeed an appropriate mixture distribution can be identified that matches the empirical data well.

5.3 Distribution Fitting for Payloads of FPS Games

In the next sections, the results of fitting the Gaussian, Log-Normal, Generalised Extreme Value and Ex-Gaussian distributions to the payloads for the 2-player games are presented for the seven FPS games. The characteristics of these distributions are outlined in Section 3.3. The parameters of the fitted distributions have been estimated by using a MLE approach; refer to Section 3.5.2, using the MATLAB
5.3 Distribution Fitting for Payloads of FPS Games

Table 5.1: Distribution Parameters for C2S Payload Length

<table>
<thead>
<tr>
<th>Game</th>
<th>EG</th>
<th>GEV</th>
<th>LogN</th>
<th>Gaussian</th>
<th>EV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>µ</td>
<td>σ</td>
<td>τ</td>
<td>shape</td>
<td>scale</td>
</tr>
<tr>
<td>Quak3 Raw</td>
<td>32.79</td>
<td>3.26</td>
<td>0.10</td>
<td>0.35</td>
<td>3.32</td>
</tr>
<tr>
<td>ETPro Raw</td>
<td>29.60</td>
<td>1.06</td>
<td>2.74</td>
<td>0.06</td>
<td>2.04</td>
</tr>
<tr>
<td>ETPro Sanitised</td>
<td>29.6</td>
<td>1.07</td>
<td>2.70</td>
<td>0.05</td>
<td>2.03</td>
</tr>
<tr>
<td>HL2DM Raw</td>
<td>50.54</td>
<td>0.46</td>
<td>5.36</td>
<td>0.16</td>
<td>9.83</td>
</tr>
<tr>
<td>HL2CS Raw</td>
<td>49.71</td>
<td>9.89</td>
<td>5.58</td>
<td>0.22</td>
<td>9.60</td>
</tr>
<tr>
<td>HL2DM Raw</td>
<td>45.76</td>
<td>6.11</td>
<td>1.38</td>
<td>0.22</td>
<td>9.60</td>
</tr>
<tr>
<td>HL2CS Raw</td>
<td>35.06</td>
<td>8.11</td>
<td>7.91</td>
<td>0.07</td>
<td>5.31</td>
</tr>
<tr>
<td>Quak3 Raw</td>
<td>47.08</td>
<td>8.57</td>
<td>13.29</td>
<td>0.02</td>
<td>12.67</td>
</tr>
<tr>
<td>ETPro Raw</td>
<td>51.98</td>
<td>22.20</td>
<td>18.19</td>
<td>0.05</td>
<td>22.11</td>
</tr>
<tr>
<td>ETPro Sanitised</td>
<td>53.63</td>
<td>20.78</td>
<td>17.91</td>
<td>0.09</td>
<td>24.09</td>
</tr>
<tr>
<td>HL2DM Raw</td>
<td>60.30</td>
<td>9.65</td>
<td>34.49</td>
<td>0.17</td>
<td>22.30</td>
</tr>
<tr>
<td>HL2CS Raw</td>
<td>46.36</td>
<td>9.65</td>
<td>34.49</td>
<td>0.17</td>
<td>22.30</td>
</tr>
<tr>
<td>Quak3 Raw</td>
<td>70.59</td>
<td>23.96</td>
<td>19.27</td>
<td>0.06</td>
<td>26.70</td>
</tr>
</tbody>
</table>

Statistics Toolbox. In the case of the Ex-Gaussian distribution the MATLAB toolbox provided by Van Zandt [134] was used. Some of the traces were sanitised to remove any outliers and periodic components and they were truncated both at the beginning and end (refer to Section 4.2.2) to ensure that only the traffic during the game-play phase was retained. The details of the sanitising of the data are given when this has occurred. The best-fit distribution was determined by inspecting the distribution (CDF), the Quantile-Quantile Plots if necessary, and the size of the λ² discrepancy measure, (refer to Section 3.7). The parameters of the fitted distributions for the client-to-server packets are presented in Table 5.1 and those for the server-to-client packets are presented in Table 5.2. This methodology is similar to that used by Borella [16] and others [85, 87, 88] when fitting distributions to FPS game empirical data-sets.

Table 5.2: Distribution Parameters for S2C Payload Length

<table>
<thead>
<tr>
<th>Game</th>
<th>EG</th>
<th>GEV</th>
<th>LogN</th>
<th>Gaussian</th>
<th>EV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>µ</td>
<td>σ</td>
<td>τ</td>
<td>shape</td>
<td>scale</td>
</tr>
<tr>
<td>Quak3 Raw</td>
<td>47.08</td>
<td>8.57</td>
<td>13.29</td>
<td>0.02</td>
<td>12.67</td>
</tr>
<tr>
<td>ETPro Raw</td>
<td>51.98</td>
<td>22.20</td>
<td>18.19</td>
<td>0.05</td>
<td>22.11</td>
</tr>
<tr>
<td>ETPro Sanitised</td>
<td>53.63</td>
<td>20.78</td>
<td>17.91</td>
<td>0.09</td>
<td>24.09</td>
</tr>
<tr>
<td>HL2DM Raw</td>
<td>60.30</td>
<td>9.65</td>
<td>34.49</td>
<td>0.17</td>
<td>22.30</td>
</tr>
<tr>
<td>HL2CS Raw</td>
<td>46.36</td>
<td>9.65</td>
<td>34.49</td>
<td>0.17</td>
<td>22.30</td>
</tr>
<tr>
<td>HL2DM Raw</td>
<td>46.36</td>
<td>9.65</td>
<td>34.49</td>
<td>0.17</td>
<td>22.30</td>
</tr>
<tr>
<td>HL2CS Raw</td>
<td>44.14</td>
<td>5.46</td>
<td>19.43</td>
<td>0.18</td>
<td>12.41</td>
</tr>
<tr>
<td>HL2CS Raw</td>
<td>41.55</td>
<td>3.97</td>
<td>14.02</td>
<td>0.21</td>
<td>8.99</td>
</tr>
<tr>
<td>Quak3 Raw</td>
<td>70.59</td>
<td>23.96</td>
<td>19.27</td>
<td>0.06</td>
<td>26.70</td>
</tr>
</tbody>
</table>

5.3.1 Client-to-Server Payload Size

This section presents the results of fitting the various distributions to the payload length of the client-to-server packets. From the results of the previous chapter, it was shown that the payload size is small and is independent of the number of clients participating in the game. The client-to-server payload size statistics for each game are presented in Table 5.3.
### Table 5.3: Statistics for C2S Payload Size

<table>
<thead>
<tr>
<th>Game</th>
<th>Min (B)</th>
<th>Max (B)</th>
<th>Mean</th>
<th>Var</th>
<th>EB</th>
<th>GEV</th>
<th>LogN</th>
<th>Gaussian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake3 Raw</td>
<td>25</td>
<td>45</td>
<td>32.9</td>
<td>10.6</td>
<td>0.439</td>
<td>0.443</td>
<td>0.981</td>
<td>0.439</td>
</tr>
<tr>
<td>Quake3 Sanitised</td>
<td>25</td>
<td>41</td>
<td>32.9</td>
<td>10.6</td>
<td>0.300</td>
<td>0.260</td>
<td>0.330</td>
<td>0.300</td>
</tr>
<tr>
<td>ETPro Raw</td>
<td>27</td>
<td>47</td>
<td>32.3</td>
<td>7.7</td>
<td>0.739</td>
<td>0.491</td>
<td>0.595</td>
<td>0.887</td>
</tr>
<tr>
<td>ETPro Sanitised</td>
<td>27</td>
<td>45</td>
<td>32.3</td>
<td>7.4</td>
<td>0.667</td>
<td>0.400</td>
<td>0.556</td>
<td>0.826</td>
</tr>
<tr>
<td>HL2DM Raw</td>
<td>10</td>
<td>102</td>
<td>55.9</td>
<td>117.5</td>
<td>2.150</td>
<td>1.790</td>
<td>1.782</td>
<td>2.339</td>
</tr>
<tr>
<td>HL2DM Sanitised</td>
<td>34</td>
<td>102</td>
<td>55.9</td>
<td>117.3</td>
<td>2.132</td>
<td>1.740</td>
<td>1.740</td>
<td>2.326</td>
</tr>
<tr>
<td>HL2CS Raw</td>
<td>33</td>
<td>90</td>
<td>51.2</td>
<td>100.1</td>
<td>3.743</td>
<td>4.268</td>
<td>4.892</td>
<td>3.733</td>
</tr>
<tr>
<td>HL2CS Sanitised</td>
<td>31</td>
<td>72</td>
<td>47.1</td>
<td>39.3</td>
<td>0.135</td>
<td>0.080</td>
<td>0.123</td>
<td>0.138</td>
</tr>
<tr>
<td>HLCS Raw</td>
<td>31</td>
<td>97</td>
<td>43.2</td>
<td>49.8</td>
<td>0.392</td>
<td>0.398</td>
<td>0.407</td>
<td>0.428</td>
</tr>
<tr>
<td>HLCS Sanitised</td>
<td>31</td>
<td>75</td>
<td>43.2</td>
<td>49.6</td>
<td>0.392</td>
<td>0.397</td>
<td>0.408</td>
<td>0.429</td>
</tr>
<tr>
<td>Quake4 Raw</td>
<td>19</td>
<td>83</td>
<td>53.9</td>
<td>79.8</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Quake4 Sanitised Lower</td>
<td>42</td>
<td>52</td>
<td>48.2</td>
<td>2.8</td>
<td>0.034</td>
<td>0.035</td>
<td>0.030</td>
<td>0.036</td>
</tr>
<tr>
<td>Quake4 Sanitised Upper</td>
<td>53</td>
<td>83</td>
<td>64.3</td>
<td>16.7</td>
<td>0.113</td>
<td>0.373</td>
<td>0.187</td>
<td>0.113</td>
</tr>
</tbody>
</table>

#### 5.3.1.1 Wolfenstein Enemy Territory Pro

The client-to-server payload size density and distributions for this game are shown in Figure 5.1. Examination of the raw data reveals that a 47 byte packet is sent periodically (approximately every 500 or so packets). The sanitised trace was constructed by removing this packet size (relative frequency $\approx 0.15\%$, 130 packets from 85049). If required, for example in a simulation, this particular payload size can be modelled by a degenerate distribution. For the sanitised data, it is evident that the Generalised Extreme Value distribution provides the best fit, as this has the lowest $\lambda^2$ discrepancy, refer to Table 5.3. Inspection of the Quantile-Quantile Plots (Figure 5.2) reveals that the Ex-Gaussian and Generalised Extreme Value distributions better fit the lower tail of the empirical distribution, whilst the Log-Normal provides a better fit to the upper tail (>40 bytes). As there is only a small percentage of the total packets in the upper tail, it can be concluded that in this case the Generalised Extreme Value distribution provides the best fit to the empirical distribution for the client-to-server payload size.

#### 5.3.1.2 Counter-Strike-Source

The client-to-server payload size density and distributions for Counter-Strike-Source are shown in Figure 5.3. From these results we see that the Log-Normal distribution provides the poorest fit, whilst the Ex-Gaussian and Gaussian result in the best fit (refer to Table 5.3). This is not unexpected, as the Gaussian distribution is a special case of the Ex-Gaussian distribution.
5.3 Distribution Fitting for Payloads of FPS Games

Figure 5.1: Client-to-Server Payload Size Density and Distribution for Wolfenstein Enemy Territory Pro

Figure 5.2: Client-to-Server Payload Size Q-Q plots for Wolfenstein Enemy Territory Pro (Sanitised Data)

Figure 5.3: Client-to-Server Payload Size Density and Distribution for Counter-Strike-Source
5.3.1.3 Half-Life 2

The client-to-server payload size density and distributions for this game are shown in Figure 5.4. In this case, the sanitised trace has had a number of small 10 byte packets removed. These packets consist of less than 10^{-3}\% of the total packets in the trace. Examination of the distribution of the sanitised data (Figure 5.4) suggests that the Log-Normal does not fit the upper tail of the empirical data well. This is also evident from the Quantile-Quantile Plots (Figure 5.5). Having said this, the $\chi^2$ discrepancy (Table 5.3) is lowest for the Log-Normal and the Generalised Extreme Value distribution. Once again, the Generalised Extreme Value distribution gives the best fit to the empirical data, but there is little to choose between these last two distributions.
5.3 Distribution Fitting for Payloads of FPS Games

5.3.1.4 Half-Life Counter-Strike

The client-to-server payload size density and distributions for this game are shown in Figure 5.6. A single packet of 97 bytes was removed in the sanitised trace. This packet size was removed as it was considered an outlier.

In this case, it is extremely difficult to choose between the distributions. The $\lambda^2$ discrepancy is similar for all the distribution, refer to (Table 5.3). Inspection of the distribution (Figure 5.6) suggests that the Generalised Extreme Value and the Ex-Gaussian distributions fit the lower tail better, whilst the Gaussian and Log-Normal fit the upper tail better. The Quantile-Quantile Plots (Figure 5.7) support this observation. For this case, the fit is difficult because of the large relative frequency of a certain payload sizes (approximately 40 bytes). A bimodal distribution could possibly provide a better fit to the trace; however, this was not pursued further.

5.3.1.5 Half-Life

The client-to-server payload size density and distributions for this game are shown in Figure 5.8. As with the case for Counter-Strike-Source, there is little difference in the fit of the various distributions. However, the Generalised Extreme Value distribution has the lowest $\lambda^2$ discrepancy, (Table 5.3).

5.3.1.6 Quake III Arena

The client-to-server payload size density and distributions for Quake III Arena are shown in Figure 5.9. The trace was sanitised by removing packets that were larger
than 41 bytes (0.01% of the packets in the trace). Again, there is very little difference in the fit for the four distributions; however, the Generalised Extreme Value distribution has the lowest $\lambda^2$ discrepancy (Table 5.3). This finding is supported by the different models that exist in the literature for this parameter, e.g. Park et al. [102] where they find that a Log-Normal is the best-fit distribution and Lang et al. [88] who find that the Gaussian is the best-fit distribution.

### 5.3.1.7 Quake IV

For Quake IV it can be seen from the density of the payload size, shown in Figure 5.10, that the distribution has three modes. The lowest mode is due to periodic packets whose lengths are approximately 25 bytes. The client sends these small packets to the server approximately once every 34 packets. These have been removed.
5.3 Distribution Fitting for Payloads of FPS Games

Figure 5.9: Client-to-Server Payload Size Density and Distribution for Quake III Arena

Figure 5.10: Client-to-Server Payload Size Density and Distribution for Quake IV

from the sanitised trace. The remaining part of the distribution is clearly bi-modal and so it has been separated into two distributions (refer to Figure 5.10). For simplicity, the trace has been separated at payload sizes of 53 bytes, but a better approach would be to fit a mixture distribution (possibly Gaussian) to this data.

Again, it is difficult to determine the best fitting distribution. For the upper part of the distribution, all but the Generalised Extreme Value distribution provide the best fit, whilst the lower part there is no clear best-fit distribution, with all four distributions having similar $\lambda^2$ discrepancy measure (Table 5.3).

5.3.1.8 Discussion of distribution fitting to the client-to-server payload

A summary of the results of fitting the four distributions are presented in Table 5.4. From these results presented, it can be seen that it is difficult to fit one particular distribution to the client-to-server payload size; however, the Generalised Extreme Value or the Ex-Gaussian distributions cannot be discounted as good candidates for
modelling this parameter. In general, there is little difference in the $\lambda^2$ discrepancy for the four distributions studied. This may be due to the fact the there is little variation in the range of payload sizes. As the data is discrete, fitting discrete distributions may lead to better demarcation in terms of finding a best-fit distribution. This possibility can be explored further; however, as the distribution of the payload size does not vary significantly, we decided not to pursue the modelling further at this stage.

### 5.3.2 Server-to-Client Payload

The results of fitting the four distributions to the server-to-client payload size for each game are presented in Table 5.5. The server-to-client payload has a larger variation in size than does that of the client-to-server payload size and the distributions of this parameter are often very skewed. There is also a relationship between the number of players and the mean payload length; this is discussed in more detail in Chapter 7.

#### 5.3.2.1 Wolfenstein Enemy Territory Pro

The server-to-client density and distributions of the payload size for this game are shown in Figure 5.11. A deeper examination of the density and the spectrum (Figure 4.9) of the raw trace reveals that there is a large number of packets with sizes less than 21 bytes, most of these are due to periodic packets that are sent to the client. The sanitised data for this trace was obtained by removing these 21 byte packets (approximately 3% of the packets). Fitting the distributions to the raw data, it can be seen that the Ex-Gaussian, Gaussian, and possibly the Generalised Extreme Value provide the best fit as they have the lowest $\lambda^2$ metrics (Table 5.5). Examination of the Quantile-Quantile Plots (Figure 5.12) reveals that the Generalised Extreme

<table>
<thead>
<tr>
<th>Game</th>
<th>EG</th>
<th>GEV</th>
<th>LogN</th>
<th>Gaussian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake3</td>
<td>Good</td>
<td>Best</td>
<td>Good</td>
<td>-</td>
</tr>
<tr>
<td>ETPro</td>
<td>Good</td>
<td>-</td>
<td>-</td>
<td>Good</td>
</tr>
<tr>
<td>HL2DM</td>
<td>-</td>
<td>Best</td>
<td>Good</td>
<td>-</td>
</tr>
<tr>
<td>HL2CS</td>
<td>-</td>
<td>No clear best distribution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLDM</td>
<td>No clear best distribution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLCS</td>
<td>-</td>
<td>Best</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Quake4</td>
<td>No clear best distribution</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.3 Distribution Fitting for Payloads of FPS Games

Table 5.5: Statistics for the S2C Payload Length

<table>
<thead>
<tr>
<th>Game</th>
<th>Min (B)</th>
<th>Max (B)</th>
<th>Mean</th>
<th>Var</th>
<th>EG</th>
<th>GEV</th>
<th>LogN</th>
<th>Gaussian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake3 Raw</td>
<td>19</td>
<td>275</td>
<td>60.5</td>
<td>328.8</td>
<td>0.099</td>
<td>0.141</td>
<td>0.160</td>
<td>0.638</td>
</tr>
<tr>
<td>Quake3 Sanitised</td>
<td>23</td>
<td>200</td>
<td>60.4</td>
<td>301.4</td>
<td>0.094</td>
<td>0.133</td>
<td>0.151</td>
<td>0.533</td>
</tr>
<tr>
<td>ETPro Raw</td>
<td>18</td>
<td>495</td>
<td>70.2</td>
<td>847.5</td>
<td>0.072</td>
<td>0.092</td>
<td>0.148</td>
<td>0.066</td>
</tr>
<tr>
<td>ETPro Sanitised</td>
<td>21</td>
<td>495</td>
<td>71.7</td>
<td>793.2</td>
<td>0.081</td>
<td>0.083</td>
<td>0.119</td>
<td>0.082</td>
</tr>
<tr>
<td>HL2DM Raw</td>
<td>39</td>
<td>1239</td>
<td>95.9</td>
<td>2096.3</td>
<td>0.331</td>
<td>0.339</td>
<td>0.519</td>
<td>1.646</td>
</tr>
<tr>
<td>HL2DM Sanitised</td>
<td>39</td>
<td>400</td>
<td>94.8</td>
<td>1510.2</td>
<td>0.159</td>
<td>0.166</td>
<td>0.302</td>
<td>0.958</td>
</tr>
<tr>
<td>HL2CS Raw</td>
<td>32</td>
<td>1165</td>
<td>67.6</td>
<td>2825.5</td>
<td>0.823</td>
<td>0.948</td>
<td>0.782</td>
<td>0.713</td>
</tr>
<tr>
<td>HL2CS Sanitised</td>
<td>33</td>
<td>248</td>
<td>66.5</td>
<td>592.8</td>
<td>0.363</td>
<td>0.474</td>
<td>0.604</td>
<td>0.732</td>
</tr>
<tr>
<td>HL2CS Aggressive</td>
<td>33</td>
<td>248</td>
<td>77.8</td>
<td>724.6</td>
<td>0.372</td>
<td>0.409</td>
<td>0.411</td>
<td>0.465</td>
</tr>
<tr>
<td>HLDM Raw</td>
<td>27</td>
<td>752</td>
<td>63</td>
<td>855.5</td>
<td>0.101</td>
<td>0.048</td>
<td>0.560</td>
<td>3.307</td>
</tr>
<tr>
<td>HLDM Sanitised</td>
<td>27</td>
<td>250</td>
<td>63.9</td>
<td>581.7</td>
<td>0.070</td>
<td>0.071</td>
<td>0.374</td>
<td>1.235</td>
</tr>
<tr>
<td>HLCS Raw</td>
<td>27</td>
<td>504</td>
<td>53.4</td>
<td>595.8</td>
<td>1.570</td>
<td>1.639</td>
<td>3.241</td>
<td>3.301</td>
</tr>
<tr>
<td>HLCS Sanitised</td>
<td>35</td>
<td>244</td>
<td>55.6</td>
<td>291</td>
<td>0.104</td>
<td>0.110</td>
<td>0.294</td>
<td>0.501</td>
</tr>
<tr>
<td>Quake4 Raw</td>
<td>25</td>
<td>412</td>
<td>81.6</td>
<td>1316.9</td>
<td>0.590</td>
<td>0.636</td>
<td>0.704</td>
<td>0.562</td>
</tr>
<tr>
<td>Quake4 Sanitised</td>
<td>35</td>
<td>250</td>
<td>89.9</td>
<td>924</td>
<td>0.797</td>
<td>0.927</td>
<td>1.231</td>
<td>0.740</td>
</tr>
</tbody>
</table>

In this case, the Generalised Extreme Value, the Ex-Gaussian and the Gaussian distributions result in the lowest $\chi^2$ discrepancy metrics (Table 5.5). These results show that the Ex-Gaussian distribution provides a good fit to both the raw and sanitised payload size. The Generalised Extreme Value distribution also provides a good fit to the sanitised data; however, at this stage there is no compelling reason to prefer the Generalised Extreme Value to the Ex-Gaussian distribution for this game.
5.3.2.2 Counter-Strike-Source

The server-to-client payload size density and distributions for this game are shown in Figure 5.13. The sanitised data in this case was obtained by retaining only those packets whose size was in the range of 33 to 248 bytes (approximately 92% of the packets); see Figure 5.14. In Section 4.4.4.1, it was shown that there was periodic traffic in this trace, so a more aggressive sanitising of the raw data was applied. In this case, all the large peaks in the density were removed (approximately 47% of all packets) from the trace, resulting in the density shown in Figure 5.14 ('aggressive'). This was done as a way of checking whether the large peaks were biasing the fit. In this case, it may be appropriate that degenerate distributions be used to model the large spikes that are found in the empirical distribution.

For all three sanitising cases, the skewed distributions provide the best fit, thus the Gaussian is not an appropriate distribution for this case. The Ex-Gaussian and Generalised Extreme Value distributions provide similar fits for the three cases; the Log-Normal provides a good fit to the raw data. Overall, the Ex-Gaussian has the lowest $\lambda^2$ discrepancy (Table 5.5).
5.3 Distribution Fitting for Payloads of FPS Games

5.3.2.3 Half-Life 2

The density and distributions of the server-to-client payload size for this game are shown in Figure 5.15. For the sanitised trace, packets of size greater than 400 bytes were removed. These packets accounted for approximately 1.6% of the total packets. Again, the skewed distributions provide the best fit both the raw and sanitised data. For this case, the Ex-Gaussian and Generalised Extreme Value distributions provide the best fit (see Figure 5.15 and Table 5.5).

5.3.2.4 Half-Life Counter-Strike

The server-to-client payload size density and distributions for this game are shown in Figure 5.16. For the sanitised trace, only packets of size ranging from 35 to 250
bytes were retained. Approximately 14% of the total packets were removed most of which were 32 bytes in length. If required, these should be modelled by a degenerate distribution. In the case of the more aggressive sanitising, the large peaks in the density were removed from the trace (approximately 45% of all packets). Again, the skewed distributions provide the best fit for both the raw and sanitised data. In this case, the Ex-Gaussian and Generalised Extreme Value provide the best fit (lowest $\lambda^2$ metric, see Table 5.5). For the data that has been aggressively sanitised, the Log-Normal distribution has a $\lambda^2$ metric that is similar to Ex-Gaussian and Generalised Extreme Value distributions. However, close inspection of the distribution (Figure 5.17) and the Quantile-Quantile Plots (Figure 5.18) seem to suggest that the Ex-Gaussian and Generalised Extreme Value provide a closer fit to the empirical data.
5.3 Distribution Fitting for Payloads of FPS Games

Figure 5.17: Server-to-Client Payload Size Density and Distribution for Half-Life Counter-Strike (Sanitised Data)

Figure 5.18: Server-to-Client Payload Size Q-Q plots for Half-Life Counter-Strike (Aggressively Sanitised Data)

5.3.2.5 Half-Life

The server-to-client payload size density and distributions for this game are shown in Figure 5.19. Packets of sizes greater than 250 bytes were removed for the sanitised trace. Approximately 12% of the total packets were discarded. In the case of the more aggressive sanitising, the packet sizes corresponding to the large peaks in the density were removed from the trace (approximately 37% of all packets). Again, these results are similar to Half-Life Counter-Strike where the Ex-Gaussian and Generalised Extreme Value distributions provide the best fit for the raw and sanitised data. In the case of the aggressively sanitised data, the Log-Normal also
Figure 5.19: Server-to-Client Payload Size Density and Distribution for Half-Life provides a good fit. Previously Lang et al. [87] fitted a Log-Normal distribution to this parameter; however, their trace was obtained from a game with more than two players.

5.3.2.6 Quake III Arena

The server-to-client packet size density for Quake III Arena is shown in Figure 5.20. The sanitised trace contains only packets of size ranging from 23 to 200 bytes. Only a small number (approximately 0.2%) of the total packets were removed. For Quake III Arena the three skewed distributions provide the best fit, both in terms of the visual inspection of the distribution (Figure 5.20) and the size of the $\chi^2$ discrepancy (Table 5.5). The Generalised Extreme Value provides a better fit to the upper tail of the empirical distribution, whilst the Ex-Gaussian distribution has the lowest discrepancy. This is in contrast to the results obtained by Lang et al. [88] where they found that a Log-Normal distribution provided the best fit.

5.3.2.7 Quake IV

The packet size density is shown in Figure 5.21. A Quake IV server periodically sends small packets of less than 30 bytes to the client (refer to Section 4.4.7.1 and Figure 4.36). These periodic packets have been removed from the sanitised trace. They account for approximately 14% of the total packets. Again, these should be modelled separately by degenerate distributions as required.
5.3 Distribution Fitting for Payloads of FPS Games

Figure 5.20: Server-to-Client Payload Size Density and Distribution for Quake III Arena

Both the raw and sanitised Quake IV traces are multi-modal thus, it is difficult to identify a single best-fit distribution. The Gaussian and Ex-Gaussian distributions have the lowest $\lambda^2$ metric (Table 5.5) for both the sanitised and raw empirical traces. Of the skewed distributions, the Ex-Gaussian and Generalised Extreme Value have similar fits.

5.3.3 Discussion of distribution fitting to the server-to-client payload

A summary of the results of fitting the four distributions to the server-to-client payload is presented in Table 5.6. Unlike the Gaussian and Log-Normal distributions, the shape of the Generalised Extreme Value and Ex-Gaussian distributions can be
adjusted to allow a continuous range of possible shapes, because of this, they provide good fits to the empirical data irrespective of whether these data are symmetrical or skewed. The Extreme Value distribution has been a popular choice to describe the payload distribution of FPS game traffic. The results of this chapter show that as there is not a great difference between the Generalised Extreme Value and the Ex-Gaussian in terms of their fit to the data, it is plausible other distributions could be used as suitable models. Extreme Value distributions arise from phenomena where either the largest or the smallest value values are observed from a large set of iid random values [58]. Is this a reasonable explanation of the underlying process that gives rise to the payload distribution? The Ex-Gaussian, on the other hand, arises from the sum of a Gaussian and an Exponentially distributed random value (a mixture distribution). Is this a better explanation of the underlying process? Further investigation of the source code for Quake III Arena indeed shows that the server-to-client packet is constructed based on the aggregation of the changes in the state of the individual game entities close to the player [118]. Whilst this evidence is based on the code from only one FPS game, it is plausible that other games would have a similar process in terms of constructing the game state update information. Certainly, the concept of the game state being derived from the sum of the state of other game entities seems a plausible general model. Given this last finding and the evidence from the empirical data, it can be asserted that the Ex-Gaussian distribution is as good a candidate to model the server-to-client payload length as is the Generalised Extreme Value (or the Extreme Value) distribution.
5.4 Summary

Many of the models of first-person shooter game payload distributions found in the literature are based on the Extreme Value distribution. In this chapter, four (Gaussian, Log-Normal, Generalised Extreme Value and the Ex-Gaussian) distributions have been fitted to the payload sizes for two-player games of seven popular FPS games to determine if other distributions are as useful for modelling. In the client-to-server direction it has been shown that the payload size can be suitably modelled, after removing the ‘spikes’, by one or more of the four distributions presented in this chapter. In several cases, there is little difference in the $\lambda^2$ metric for the fit of the four distributions. This is attributed to the small variation in the payload size distribution. On the other hand, the Generalised Extreme Value and Ex-Gaussian distributions are superior in terms of their fits to the server-to-client packet empirical data for most of the games. Both these distributions are versatile as their shape can be adjusted to fit the data. In much of the literature, the Extreme Value distribution has been used to model FPS packet size distributions without any justification as to why the payloads would be distributed in this manner. In this chapter, it has been shown that another skewed distribution (Ex-Gaussian) also models the observed data just as well as the Extreme Value distribution.

The shortcomings of drawing random samples from a standard distribution are that individual distributions need to be fitted to each of the game scenarios to be modelled. Thus, a large number of controlled experiments need to be conducted in order to gather the data required to build the models. This last approach also suffers from the fact that as the models are tied to the empirical measurements of a particular game session, they may not be general enough. What is required is a method of scaling the models in such a way that they can be used to model the characteristics of games with larger numbers of players. This issue will be addressed in Section 7.2, where methods enabling extrapolation of these models are presented.

Another shortcoming of this method is that, as mentioned previously, these models do not incorporate any serial correlation that may exist in the payload sizes. In other words, these models assume that the packets’ payload sizes are independent. In Chapter 4, it was shown that there is significant autocorrelation in the payload size of the server-to-client update packet; this issue will addressed in the next chapter.
The main findings of this chapter are that the server-to-client can be modelled using an Ex-Gaussian distribution, this is a novel contribution. The use of this distribution addresses the important question posed in this research, specifically: “What types of statistical distributions are suitable for modelling FPS traffic parameters?” (Section 1.4).
Chapter 6

Time Series Modelling

6.1 Introduction

Much of the literature on the characterisation of multiplayer FPS game traffic is
deals with the inter-arrival time and payload size distributions of the update pack-
ets. However, the way in which the size of the packet changes from one packet to
the next should be included in a comprehensive traffic model. As was shown in
Section 2.9 some traffic models of FPS game traffic derived from empirical data, for
example Lang et al. [87], have actually assumed that the correlation in the size of
successive packets in the traffic is negligible. According to Branch and Armitage [19],
knowledge of the correlation structure of the FPS traffic is useful in understanding
the effects on delay and jitter when mixing this traffic with other types of traffic.
Therefore, it is important that this characteristic be included in a model, if realistic
simulation models are to be implemented. The results of Chapter 4, show that there
is considerable autocorrelation in the server-to-client payload packet length, which
contradicts the assumption above. In this chapter, an investigation of modelling
this correlation using time-series analysis techniques is carried out.

The aim of this chapter is to determining a suitable model for the server-to-
client packet length autocorrelation for FPS games during the game-play phase.
The traffic generated during other phases of game play is not considered. The aim
of this analysis is to obtain as simple a time-series model as possible, which can
capture the distribution of the empirical packet size for the various games, while
retaining the correlation structure.

In Chapter 4, it was found that the client-to-server payload showed little variation
in size, in contrast, the server-to-client payload length showed considerable variation
and that in general its ACVF decayed exponentially. Given this observation, a time-
series model seems appropriate for this situation.

In this chapter, an analysis of the time-series behaviour of game traffic (a weak
area of the current models in the literature) is presented. Initially the AR(1) model
is investigated and the shortcomings of this simple model are exposed. The devel-
opment of a more suitable time-series model, the ARMA(1,1), for the S2C payload
length for FPS games is the main contribution of this chapter. Parts of the work
in this chapter have been published in Cricenti and Branch [40], Cricenti et al. [42]
and Cricenti and Branch [43].

The rest of the chapter is organized as follows. Section 6.2 presents results from
fitting time-series models to the server-to-client packet traffic. Section 6.3 presents
conclusions.

6.2 Time-series Modelling of FPS Games

In this section, measurements obtained from real traffic traces are used to investigate
which time-series models are suitable for modelling the server-to-client payload size
of the traffic produced by the seven FPS games. With parsimony in mind, initially
the simplest models, (the AR(p) and MA(q)), are investigated to determine if they
are suitable. The simple ARMA models are also examined, if these are adequate
then there is no need to consider the more complicated models. If necessary in the
cases where the ARMA model is inadequate, such as when a trend is present, the
ARIMA or FARIMA models can be used.

The aim of this section is to obtain as simple a time-series model as possible that
will capture the variance and density of the empirical server-to-client payload size
for the different games, while retaining the correlation structure. The Parsimony
principle will be used to aid the choice of the simplest model that gives an adequate
match for the variance of the process.

6.2.1 Models for the S2C Payload for FPS games

The following methodology, developed based on Box et al. [17] and Branch et al.
[21] and published in [43], was used to fit the time-series models to the traffic traces:

- The traffic traces for the games are truncated as per Chapter 4 section 4.2.2
to ensure that only the game-play phase is modelled.
Where trends or periodic components are present in the trace, the effects of these are examined, and if necessary, these are removed to ensure that the trace data are stationary. If an ARIMA model is required then this step can be omitted.

The time-series that represent the S2C payload are characterised by the parameters $\Phi(B)$ (equation 3.81), $\Theta(B)$ (equation 3.82), the residuals $\varepsilon_n$ and the variance of residuals $\sigma^2$ (refer to equation 3.80). If necessary, the differencing parameter $d$ can also be estimated. This is required when the trace is not stationary. These parameters are estimated using the ‘R’ statistical package [111].

The ACVF and the PACF for the model are used to aid in determining the type and order of the model [17].

The residuals $\varepsilon_n$ of the process are used to drive the model and thus create a synthetic S2C payload size trace. The residuals are whitened by shuffling them to ensure that they are not correlated.

The PDF of the empirical and the synthetic trace are compared visually for the accuracy of the fit.

Higher order models are checked and the AIC is used to find the most parsimonious model. The $\lambda^2$ could be used for the comparison, but this was not used in this part of the work.

For the FPS games in this study, analysis of the traces revealed that there were no significant drifts (trends) and that the differencing parameter $d$ was approximately zero, thus it is not necessary to consider the more general FARIMA$(p,d,q)$ or ARIMA$(p,d,q)$ models. Where the traces showed evidence of periodicities, these were removed. Details of this are discussed where appropriate. In the rest of this discussion, the sequence of observations $X_n$ represents the payload size of the server-to-client update packet, where $n$ denotes the packet number.

### 6.2.2 Server-to-Client First Order AR(1) Models

Branch et al. [22] argued that the correlation between successive packets of FPS games could be modelled by a Markov chain. In this case, the server-to-client payload size of the current packet depends only on that of the most recent packet sent. This assertion is based on the fact that the ACVF dies off exponentially, Chapter 4 shows some examples of the ACVFs for various FPS games. These results suggest that a first order AR(1) Markov model, one of the two simplest models (the
Figure 6.1: AR() model driven by the residuals sequence

Table 6.1: S2C payload AR(1) $\phi_1$ Parameter

<table>
<thead>
<tr>
<th>Players</th>
<th>ETPro</th>
<th>HL2DM</th>
<th>HL2CS</th>
<th>HLDM</th>
<th>HLCS</th>
<th>Quake3</th>
<th>Quake4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.73</td>
<td>0.66</td>
<td>0.74</td>
<td>0.34</td>
<td>0.48</td>
<td>0.47</td>
<td>0.39</td>
</tr>
<tr>
<td>3</td>
<td>0.68</td>
<td>0.60</td>
<td>0.76</td>
<td>0.58</td>
<td>0.53</td>
<td>0.37</td>
<td>0.33</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0.63</td>
<td>0.74</td>
<td>0.59</td>
<td>0.51</td>
<td>0.36</td>
<td>0.30</td>
</tr>
<tr>
<td>5</td>
<td>0.73</td>
<td>0.57</td>
<td>0.69</td>
<td>0.41</td>
<td>0.47</td>
<td>0.32</td>
<td>0.37</td>
</tr>
<tr>
<td>6</td>
<td>0.79</td>
<td>0.61</td>
<td>0.71</td>
<td>0.40</td>
<td>0.52</td>
<td>0.29</td>
<td>0.28</td>
</tr>
<tr>
<td>7</td>
<td>0.75</td>
<td>0.53</td>
<td>0.67</td>
<td>0.37</td>
<td>0.58</td>
<td>0.30</td>
<td>0.30</td>
</tr>
<tr>
<td>8</td>
<td>0.70</td>
<td>0.58</td>
<td>0.60</td>
<td>0.36</td>
<td>0.60</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.79</td>
<td>0.57</td>
<td>0.61</td>
<td>0.43</td>
<td>0.61</td>
<td>0.29</td>
<td></td>
</tr>
</tbody>
</table>

other is the MA(1)) that could be fitted, may be appropriate as a time-series model for the server-to-client payload size.

Observation of the ACVFs of the traces, shown in Chapter 4, reveals that these do not cut off abruptly, implying that a pure MA(p) process is not a suitable model. Therefore, the MA(q) process will not be considered further.

The statistical package ‘R’ [111] was used to fit AR(1) models to each game, Table 6.1 summarizes the model parameters for the games studied. The results show that there is little variation in the $\phi_1$ parameter as the number of players for a particular game is varied. This result is pleasing, as it suggests that a common AR(1) model which is driven by a residuals sequence that depends on the particular game being synthesised could be used for simulating the server-to-client traffic for a particular FPS game (shown in Figure 6.1). Unfortunately, later in this chapter we demonstrate that this simple AR(1) model is not suitable.

In the next sections we fit AR(1) models to the traffic of the seven FPS games.

6.2.3 Server-to-Client AR(1) Models

The values of the variance of the empirical server-to-client payload size distribution and that of the AR(1) synthetic distribution for the Quake III Arena, Half-Life
6.2 Time-series Modelling of FPS Games

Table 6.2: Variance for Empirical and Synthetic AR(1) S2C payload

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake3</td>
<td>HLCS</td>
<td>HLDm</td>
<td>HL2CS</td>
<td>HLD2m</td>
<td>ETPro</td>
<td>PlaYers</td>
<td>HLCS</td>
<td>HLDm</td>
<td>HL2CS</td>
<td>HLD2m</td>
</tr>
<tr>
<td>2</td>
<td>573</td>
<td>1081</td>
<td>682</td>
<td>972</td>
<td>3718</td>
<td>2973</td>
<td>3055</td>
<td>7417</td>
<td>2902</td>
<td>6912</td>
</tr>
<tr>
<td>3</td>
<td>1903</td>
<td>1268</td>
<td>2091</td>
<td>1213</td>
<td>2475</td>
<td>4073</td>
<td>2734</td>
<td>6469</td>
<td>4097</td>
<td>9426</td>
</tr>
<tr>
<td>4</td>
<td>1823</td>
<td>1212</td>
<td>2057</td>
<td>1596</td>
<td>3797</td>
<td>2031</td>
<td>2482</td>
<td>6514</td>
<td>4185</td>
<td>10321</td>
</tr>
<tr>
<td>5</td>
<td>2233</td>
<td>2408</td>
<td>3011</td>
<td>2111</td>
<td>2471</td>
<td>3224</td>
<td>4308</td>
<td>9744</td>
<td>8360</td>
<td>14972</td>
</tr>
<tr>
<td>6</td>
<td>2956</td>
<td>3396</td>
<td>1963</td>
<td>4248</td>
<td>3174</td>
<td>9484</td>
<td>3599</td>
<td>5684</td>
<td>19282</td>
<td>22262</td>
</tr>
<tr>
<td>7</td>
<td>2956</td>
<td>3396</td>
<td>1963</td>
<td>4248</td>
<td>3174</td>
<td>9484</td>
<td>3599</td>
<td>5684</td>
<td>19282</td>
<td>22262</td>
</tr>
<tr>
<td>8</td>
<td>3694</td>
<td>4155</td>
<td>1555</td>
<td>2769</td>
<td>5295</td>
<td>4471</td>
<td>4056</td>
<td>8607</td>
<td>11741</td>
<td>23431</td>
</tr>
<tr>
<td>9</td>
<td>3694</td>
<td>4155</td>
<td>1555</td>
<td>2769</td>
<td>5295</td>
<td>4471</td>
<td>4056</td>
<td>8607</td>
<td>11741</td>
<td>23431</td>
</tr>
</tbody>
</table>

Counter-Strike, Half-Life, Counter-Strike-Source, Half-Life 2, and Wolfenstein Enemy Territory Pro games are shown in Table 6.2. We can see that for all the games the variance of the synthetic trace is much larger than that of the empirical trace. For the Counter-Strike-Source and Wolfenstein Enemy Territory Pro games, the variance of the synthetic trace is approximately twice that of the corresponding empirical trace. Examination of the empirical and synthetic densities of the traces for the AR(1) model (Figures 6.2 to 6.7) reveals that the height at the mode of the synthetic density is lower than that of the empirical density. Generally, the match for the upper tail of the density is good for some games, for example Quake III Arena (Figure 6.2), Half-Life Counter-Strike (Figure 6.3), and Half-Life (Figure 6.4). On the other hand, the lower tails of the synthetic density do not match the corresponding lower tails of the empirical distributions well. In some cases the support of the lower tail includes negative values, for example Half-Life Counter-Strike (Figure 6.3), and Half-Life (Figure 6.4). This is problematic as it results in a non-zero probability of obtaining negative packet lengths. Given these results, we conclude that the AR(1) model is not suitable and that higher order AR(p) models should be investigated for these games.

6.2.3.1 Server-to-Client AR(1) Model for Quake IV

The variance of the raw empirical server-to-client payload length distribution and that of the AR(1) synthetic distribution are shown in Table 6.3. The density for the raw trace is shown in Figure 6.8, from this, one can observe a large spike outside the main lobe of the distribution, which seems to be the cause of the mismatch in the lower tails. Section 4.4.7 showed that there was a large periodic component in the trace for this game. This periodicity is removed from the trace, to ensure stationarity and so it does not bias the results. Sanitising the trace, so that the
Figure 6.2: Quake III Arena: Typical Density of S2C Payload Size: AR(1)

Figure 6.3: Half-Life Counter-Strike: Density of S2C Payload Size: AR(1)

Figure 6.4: Half-Life: Density of S2C Payload Size: AR(1)
6.2 Time-series Modelling of FPS Games

Figure 6.5: Counter-Strike-Source: Density of S2C Payload Size: AR(1)

Figure 6.6: Half-Life 2: Density of S2C Payload Size: AR(1)

Figure 6.7: ETPro: Density of S2C Payload Size: AR(1)
Table 6.3: Quake4: Variance for Empirical and Synthetic AR(1) S2C payload

<table>
<thead>
<tr>
<th>Players</th>
<th>Empirical</th>
<th>Synthetic (raw)</th>
<th>Synthetic (clean)</th>
<th>Residuals (Raw)</th>
<th>AIC (Clean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1317</td>
<td>1654</td>
<td>1669</td>
<td>1120</td>
<td>138136</td>
</tr>
<tr>
<td>3</td>
<td>3057</td>
<td>3676</td>
<td>3594</td>
<td>2727</td>
<td>144987</td>
</tr>
<tr>
<td>4</td>
<td>4509</td>
<td>5057</td>
<td>5156</td>
<td>4110</td>
<td>152193</td>
</tr>
<tr>
<td>5</td>
<td>7346</td>
<td>9106</td>
<td>8970</td>
<td>6361</td>
<td>158846</td>
</tr>
<tr>
<td>6</td>
<td>9889</td>
<td>11043</td>
<td>11084</td>
<td>9120</td>
<td>159968</td>
</tr>
<tr>
<td>7</td>
<td>12235</td>
<td>13753</td>
<td>13804</td>
<td>11161</td>
<td>168225</td>
</tr>
</tbody>
</table>

Figure 6.8: Quake4: Density of S2C Payload Size: AR(1)

spike is removed, produces the results in shown Figure 6.9. Sanitising the trace does not improve the match for the variance or for the density, as can be seen the fit for the 2-player game is very poor. The situation improves for the trials with larger numbers of players, but the height at the mode of the synthetic density is still lower than that of the empirical trace and the variance of the synthetic trace is still higher than that of the empirical trace. Once again, we conclude that the AR(1) model is not suitable for this game.

6.2.4 Higher order AR(p) Models Server-to-Client

Higher order AR(p) models were fitted to the server-to-client payload trace for each game. Increasing the order $p$ of the AR(p) model generally results in a better match between the variance of the synthetic trace and that of the empirical trace.
In order to compare the different models, the normalised variance, which in this case is defined as the ratio of the variance of the synthetic trace for the different model orders to the variance of the empirical trace, and the normalised AIC, defined as the ratio of the AIC of the AR(p) model to the AIC of the AR(1) model are used. The variation in the normalised variance and AIC for the various AR(p) model orders are presented in Figures 6.10 to 6.15. Using the normalised AIC and the normalised variance we see that the AR(1) model is not the best choice for modelling these games, confirming what we found in the previous analysis. For the Quake III Arena, Half-Life Counter-Strike, Half-Life 2, and Wolfenstein Enemy Territory Pro games the AR(5) seems a to be a better model since the variance of the synthetic trace is comparable to that of the empirical trace and the AIC is close to its minimum value. For Half-Life a model order greater than 5 \( (p > 5) \) provides a better match. While for Counter-Strike-Source (Figure 6.13), increasing the order of the AR(p) model to \( p = 3 \) results in a better match between the variance of the synthetic trace and that of the empirical trace. However, in this case the AIC is not close to its minimum value, which occurs for \( p > 6 \).

The density of S2C payload size for various model orders is shown in Figures 6.16 to 6.21. For the Counter-Strike-Source (Figure 6.19) and Wolfenstein Enemy Territory Pro (Figure 6.21) games, the higher order models provide a superior fit to the empirical density than for the AR(1) model. For the Half-Life Counter-Strike (Figure 6.17) and Half-Life (Figure 6.18) games, the match improves for certain cases, while for Quake III Arena (Figure 6.16), the higher order models do not
appear to improve the fit and in some cases, since the variance of the distribution becomes large, the fit is worse than the AR(1) model.
Figure 6.12: Half-Life: Normalised Variance and AIC of S2C Payload Size: AR(p)

Figure 6.13: Counter-Strike-Source: Normalised Variance and AIC of S2C Payload Size: AR(p)

Figure 6.14: Half-Life 2: Normalised Variance and AIC of S2C Payload Size: AR(p)
Figure 6.15: Wolfenstein Enemy Territory Pro: Normalised Variance and AIC of S2C Payload Size: AR(p)

Figure 6.16: Quake III Arena: Density of S2C Payload Size: AR(p)

Figure 6.17: Half-Life Counter-Strike: Density of S2C Payload Size: AR(p)
6.2 Time-series Modelling of FPS Games

**Figure 6.18:** Half-Life: Density of S2C Payload Size: AR(p)

**Figure 6.19:** Counter-Strike-Source: Density of S2C Payload Size: AR(p)

**Figure 6.20:** Half-Life 2: Density of S2C Payload Size: AR(p)
6.2.4.1 Higher order AR(p) models for Quake IV

In the case of Quake IV, increasing the order of the AR(p) models results in an improvement in both the match for the variance and for the fit to the density. Examination of the normalised variance and the AIC (Figure 6.22) shows that the AR(1) model is not the best choice for this game. For this game, the AIC drops significantly as the model order $p$ is increased above one, and drops off at a slower rate for model orders above four. The AR(5) is a good model, as the variance of this model matches that of the empirical trace.

The density of server-to-client payload size for various model sizes is shown Figure 6.23. In some cases, the fit of the density is worse than the AR(1) and for some models, there is larger density of negative packet lengths. With the exception of the 2-player game, the density produced by the AR(5) model visually fits the empirical density well.

6.2.4.2 Discussion

In general, the simple AR(1) process is not a good choice to model the server-to-client payload size, as the density of the synthetic trace produced by this model is
6.2 Time-series Modelling of FPS Games

Figure 6.22: Quake IV: Normalised Variance and AIC of S2C Payload Size: AR(p) Clean

Figure 6.23: Quake IV: Density of Sanitised S2C Payload Size: AR(p) Clean

generally wider and the mode is generally lower than that of the empirical density. As a consequence, the variance of the synthetic trace is consistently approximately double that of the empirical process. For some games, the lower tail of the density of the synthetic trace extends below zero. This implies that there is a non-zero probability of packets with negative lengths. In a simulation, this is not a large problem as these packets can be discarded. However, truncating the distribution leads to a higher proportion of small packets being generated since the lower tail is removed (see Figure 6.24).

Higher order AR(p) models were tried with an AR(5) models being a good candidate for the various games. However, it is not surprising that these higher order models provide a good fit, since the higher the model order the better the fit to a specific data set. Because a parsimonious simple model is required, these higher
order AR models will not be considered, as there is the risk that the model has been over-fitted to the data. In this case, the model becomes specific to the data-set and is not generalizable.

Given these observations, one must conclude that the AR(1) model, although quite simple, does not model the empirical process well, and therefore, will not be considered further. In the next section, the ARMA case will be examined to determine if it is a suitable model.

### 6.2.5 Server-to-Client ARMA(p,q) Models

The methodology used for this section is similar to that for the AR(p) case of the previous section, with the exception that the statistical package ‘R’ [111] was used to fit ARMA(p,q) models to the empirical server-to-client traces of each game. The simplest ARMA model is desirable, as this will have properties that can be simply exploited in developing an extrapolation technique, this is further discussed in Section 7.6. The parameters of the ARMA(1,1), the simplest ARMA model, are
6.2 Time-series Modelling of FPS Games

Table 6.4: S2C ARMA(1,1) $\phi_1$, $\theta_1$ Parameter

<table>
<thead>
<tr>
<th>Players</th>
<th>ETPro</th>
<th>HL2DM</th>
<th>HL2CS</th>
<th>HLDM</th>
<th>HLCS</th>
<th>Quake3</th>
<th>Quake4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_1$</td>
<td>$\theta_1$</td>
<td>$\phi_1$</td>
<td>$\theta_1$</td>
<td>$\phi_1$</td>
<td>$\theta_1$</td>
<td>$\phi_1$</td>
<td>$\theta_1$</td>
</tr>
<tr>
<td>2</td>
<td>0.98</td>
<td>0.69</td>
<td>0.97</td>
<td>0.71</td>
<td>0.79</td>
<td>0.12</td>
<td>0.97</td>
</tr>
<tr>
<td>3</td>
<td>0.99</td>
<td>0.70</td>
<td>0.97</td>
<td>0.77</td>
<td>0.82</td>
<td>0.14</td>
<td>0.97</td>
</tr>
<tr>
<td>4</td>
<td>0.98</td>
<td>0.71</td>
<td>0.97</td>
<td>0.76</td>
<td>0.80</td>
<td>0.12</td>
<td>0.98</td>
</tr>
<tr>
<td>5</td>
<td>0.99</td>
<td>0.76</td>
<td>0.98</td>
<td>0.79</td>
<td>0.76</td>
<td>0.14</td>
<td>0.97</td>
</tr>
<tr>
<td>6</td>
<td>0.98</td>
<td>0.70</td>
<td>0.98</td>
<td>0.77</td>
<td>0.81</td>
<td>0.21</td>
<td>0.97</td>
</tr>
<tr>
<td>7</td>
<td>0.98</td>
<td>0.73</td>
<td>0.98</td>
<td>0.81</td>
<td>0.79</td>
<td>0.22</td>
<td>0.96</td>
</tr>
<tr>
<td>8</td>
<td>0.99</td>
<td>0.80</td>
<td>0.98</td>
<td>0.80</td>
<td>0.85</td>
<td>0.41</td>
<td>0.98</td>
</tr>
<tr>
<td>9</td>
<td>0.99</td>
<td>0.72</td>
<td>0.98</td>
<td>0.78</td>
<td>0.83</td>
<td>0.36</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 6.5: Variance for Empirical and Synthetic ARMA(1,1) S2C payload

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.99</td>
<td>0.97</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>3</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.97</td>
</tr>
<tr>
<td>4</td>
<td>0.97</td>
<td>0.96</td>
<td>0.97</td>
<td>0.96</td>
<td>0.97</td>
<td>0.97</td>
<td>0.96</td>
<td>0.97</td>
<td>0.96</td>
<td>0.97</td>
<td>0.96</td>
<td>0.97</td>
<td>0.96</td>
</tr>
<tr>
<td>5</td>
<td>0.96</td>
<td>0.95</td>
<td>0.96</td>
<td>0.95</td>
<td>0.96</td>
<td>0.95</td>
<td>0.95</td>
<td>0.96</td>
<td>0.95</td>
<td>0.96</td>
<td>0.95</td>
<td>0.96</td>
<td>0.95</td>
</tr>
<tr>
<td>6</td>
<td>0.95</td>
<td>0.94</td>
<td>0.95</td>
<td>0.94</td>
<td>0.95</td>
<td>0.94</td>
<td>0.94</td>
<td>0.95</td>
<td>0.94</td>
<td>0.95</td>
<td>0.94</td>
<td>0.95</td>
<td>0.94</td>
</tr>
<tr>
<td>7</td>
<td>0.94</td>
<td>0.93</td>
<td>0.94</td>
<td>0.93</td>
<td>0.94</td>
<td>0.93</td>
<td>0.93</td>
<td>0.94</td>
<td>0.93</td>
<td>0.94</td>
<td>0.93</td>
<td>0.94</td>
<td>0.93</td>
</tr>
<tr>
<td>8</td>
<td>0.93</td>
<td>0.92</td>
<td>0.93</td>
<td>0.92</td>
<td>0.93</td>
<td>0.92</td>
<td>0.92</td>
<td>0.93</td>
<td>0.92</td>
<td>0.93</td>
<td>0.92</td>
<td>0.93</td>
<td>0.92</td>
</tr>
<tr>
<td>9</td>
<td>0.92</td>
<td>0.91</td>
<td>0.92</td>
<td>0.91</td>
<td>0.92</td>
<td>0.91</td>
<td>0.91</td>
<td>0.92</td>
<td>0.91</td>
<td>0.92</td>
<td>0.91</td>
<td>0.92</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Table 6.6: AIC for AR(1) and ARMA(1,1) models

<table>
<thead>
<tr>
<th>Quake</th>
<th>Players</th>
<th>AR(1)</th>
<th>ARMA(1,1)</th>
<th>AR(1)</th>
<th>ARMA(1,1)</th>
<th>AR(1)</th>
<th>ARMA(1,1)</th>
<th>AR(1)</th>
<th>ARMA(1,1)</th>
<th>AR(1)</th>
<th>ARMA(1,1)</th>
<th>AR(1)</th>
<th>ARMA(1,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>4</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>5</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
</tr>
<tr>
<td>6</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
</tr>
<tr>
<td>7</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>8</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>106</td>
</tr>
<tr>
<td>9</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
</tr>
</tbody>
</table>

summarised in Table 6.4. Again, it is pleasing to note that the parameter values are similar for each particular FPS game, for the differing number of players. This property is discussed further in Section 7.5. In the following sections, a discussion of the ARMA models for the various games is presented.

6.2.5.1 Server-to-Client ARMA(1,1) Model

The variance of the empirical server-to-client, payload length distribution and that of the ARMA(1,1) synthetic distribution for the Quake III Arena, Half-Life Counter-Strike, Half-Life, Counter-Strike-Source, Half-Life 2, and Wolfenstein Enemy Territory Pro games are shown in Table 6.5. The results for Quake IV are discussed in the next section. For all these games and combinations of players, it is evident that the ARMA(1,1) model provides a better match between the variance of the empirical
and synthetic process than did the AR(1) model. Some typical examples of the empirical and synthetic densities are shown in Figures 6.25 to 6.30. Visual inspection of the densities of the Counter-Strike-Source (Figure 6.29), Half-Life 2 (Figure 6.30), Half-Life (Figure 6.28), Wolfenstein Enemy Territory Pro (Figure 6.26) shows the ARMA(1,1) models provide a superior fit to the empirical density than does the AR(1) model. Generally for these cases, the upper tails of the synthetic distribution match those of the empirical distribution very well and the match for the lower tail is also better than the AR(1) cases. For Quake III Arena (Figure 6.25) the density plots are similar to those produced by the AR(1) model. The upper tails of the synthetic distribution match those of the empirical distribution very well, as was the case for the AR(1) model. The match for the lower tail, although similar to the AR(p) is not as good for the 6 and 9-player games. The peak at mode is also slightly higher for this case. In the case of Half-Life Counter-Strike (Figure 6.27) are similar to those of the AR(1) model, with a very good match for the 5-player case. The peak at the mode is higher for the ARMA(1,1) case than the AR(1).

When considering higher order ARMA(p,q) models, we find that increasing the model order, does not significantly improve the match for the variance, (Figures 6.31 to 6.36).

The AIC shown in Table 6.6 is generally lower for the ARMA(1,1) model than for the AR(1) models of the previous section. Although increasing the model order above ARMA(1,1) typically reduces the AIC, this increase does not result in a large drop in many cases. These results are shown in Figures 6.31 to 6.36. In these graphs
6.2 Time-series Modelling of FPS Games

Figure 6.26: ETPro: Density of S2C Payload Size: ARMA(1,1)

Figure 6.27: Half-Life Counter-Strike: Density of S2C Payload Size: ARMA(1,1)

Figure 6.28: Half-Life: Density of S2C Payload Size: ARMA(1,1)
the first point is the normalised AIC for the AR(1) model, the second point is the ARMA(1,1) and so on.

6.2.5.2 Quake IV Server-to-Client ARMA(1,1) Model

The variance of the empirical server-to-client payload length distribution and that of the ARMA(1,1) synthetic distribution are shown in Table 6.7. From the density plot (Figure 6.37) it is evident that the large spike outside the main lobe of the distribution, should be removed as was done in the AR(1) case. Removing the ‘spike’ produces the results shown in Figure 6.38.

The density of the fit for the 2-player game is poor, due to the multi-modal nature of the density for this trial. For the sanitised trace, the fit for the density
6.2 Time-series Modelling of FPS Games

Figure 6.31: Quake III Arena: Normalised Variance and AIC of S2C Payload Size: ARMA()

Figure 6.32: Wolfenstein Enemy Territory Pro: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1)

Figure 6.33: HLCS: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1)
Figure 6.34: Half-Life: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1)

Figure 6.35: Counter-Strike-Source: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1)

Figure 6.36: Half-Life 2: Normalised Variance and AIC of S2C Payload Size: ARMA(p,1)
Table 6.7: Quake IV Variance for Empirical and Synthetic ARMA(1,1) S2C payload size

<table>
<thead>
<tr>
<th>Players</th>
<th>Empirical</th>
<th>Synthetic</th>
<th>Residuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1163</td>
<td>1181</td>
<td>674</td>
</tr>
<tr>
<td>3</td>
<td>2587</td>
<td>2681</td>
<td>1647</td>
</tr>
<tr>
<td>4</td>
<td>3559</td>
<td>3506</td>
<td>2171</td>
</tr>
<tr>
<td>5</td>
<td>7155</td>
<td>6593</td>
<td>4909</td>
</tr>
<tr>
<td>6</td>
<td>9095</td>
<td>8735</td>
<td>6415</td>
</tr>
<tr>
<td>7</td>
<td>11124</td>
<td>10849</td>
<td>7497</td>
</tr>
</tbody>
</table>

Figure 6.37: Quake IV: Density of S2C Payload Size: ARMA(1,1)

improves for the trials with larger numbers of players and the ARMA(1,1) model has a better match to the variance compared with the AR(1) model, but in general the AIC is lower.

6.2.5.3 Discussion

The simple ARMA(1,1) model is a better choice to model the server-to-client payload size than the AR(1), as the density of the synthetic trace produces a better fit to the empirical density. The variance of the synthetic trace is approximately the same as that of the empirical process. Higher order ARMA models were tried; however, neither the fit to the density nor the match for the variance improved significantly enough to justify using a higher order model. As previously mentioned, a parsimonious simple model is required, thus the ARMA(1,1) model seems to be a good choice.
6.3 Conclusions

In this chapter, we continued the investigation of the FPS traffic traces to demonstrate how to construct more complete models to those previously proposed. We have presented the results of fitting time-series models to the traces for the FPS games. Initially, the simplest time-series model was investigated as a model to describe the temporal characteristics of the server-to-client payload size. It was shown
that the AR(1) model is only moderately successful in capturing the variance of the packet lengths. Higher order AR(p) models were also tried. In many cases, these led to a better match between the empirical and synthetic traces. The ARMA(p,q) models were also evaluated. The simplest of these (ARMA(1,1)) produced synthetic traces whose variance was better matched to the variance of the empirical traces than the AR(1). For the ARMA models, increasing the model order generally lead to a lower AIC. However, in many of these cases the fit for the variance was not improved significantly. Care should be exercised with the higher order models so that these are not over-fitted to the data. Given the results of this chapter, we conclude that the ARMA(1,1) is a satisfactory simplified model. The FARIMA models were not considered, as there was no evidence of LRD behaviour in the traffic of the FPS games that were studied.

In the next chapter, the ARMA(1,1) model will be used as the basis of an extrapolation technique for the server-to-client payload size of FPS games.

Figure 6.40: Quake IV: Density of S2C Payload Size: ARMA(1,1)
Chapter 7

N-player Extrapolation Models

7.1 Introduction

In Chapter 4, we showed that the mean packet payload size increases linearly with the number of players. This result raises the question as to whether it is possible to extrapolate the statistics of an N-player game from measurements of games with fewer players. This last issue is important, since, if the models can be scaled, then there is no need to run further controlled trials for a particular game scenario in order to collect the additional data required to formulate the new model. Additionally, it was also shown in Chapter 6 that time-series modelling techniques are useful in FPS traffic modelling, as they capture the temporal characteristics of the server-to-client payload traffic. Branch and Armitage [18] developed a technique for extrapolating the N-player server-to-client traffic statistics from the traffic obtained from 2 and 3-player FPS games based on repeated convolution operations. One of the significant contributions of this chapter is the use of a ‘per player’ distribution function, which is used as the basis for the extrapolation techniques. This chapter expands on the work by Branch and Armitage [18] by introducing new extrapolation techniques that are more computationally efficient and that capture the correlation nature of FPS traffic, something that Branch and Armitage did not incorporate in their model.

The rest of the chapter is organized as follows. Section 7.2 presents a brief review the work by Branch and Armitage and a procedure for determining the number of convolutions required for a particular number of players in a game is derived. Section 7.3 presents a new extrapolation technique based on the Ex-Gaussian distribution of Section 3.4.1. Sections 7.5 to 7.8 present techniques for extrapolating the time-series models of Chapter 6. In Section 7.9, conclusions are presented.
7.2 Extrapolating Standard Distribution Models

Branch and Armitage [18] presented a technique where the statistics of the payload size in the server-to-client direction of an N-player FPS game could be determined from the empirical measurements of the statistics of the payload size of 2 and 3-player FPS games. This extrapolation technique is based on the assumptions that the nature of game play for an individual player is independent of the number of players participating in the game; that the individual players’ behaviour is similar; and that the update packets have a fixed part, and a variable part that is dependent on the player behaviour. Using these assumptions, Branch and Armitage [18] showed that the random variable describing the variable part of the payload size of the server-to-client update packets of an N-player game could be constructed by adding together the corresponding random variables of games with fewer players.

A summary of the procedure taken from Branch and Armitage [18], is presented below:

- Capture the FPS traffic during the game-play phase of games with 2 and 3 players.
- Determine the statistics and density of the payload size from the captured traffic.
- Determine the length of the fixed component of each packet by inspecting the density.
- Perform the necessary number of convolutions to predict the density of payload size for the N-player game.
- As the convolution will shift the density to the right by the fixed part of the packet length, the density must be shifted left by the same amount.

An expression for the number of convolutions required can be derived by using the following argument. In general, if $Z_N$ denotes the variable part of the payload size of an N-player game, then using the argument above it can be shown that:

$$Z_N = \sum_{i=1}^{a} Z_3^{(i)} + \sum_{i=1}^{b} Z_2^{(i)} \quad (7.1)$$

where $Z_2$ and $Z_3$ denote the random variable representing the variable part of the payload of a 2 and 3-player game respectively. The scalars $a$ and $b$ denote the number of iid random variables that must be summed. These scalars are subject to the constraint: $N = 3a + 2b$. One possible set of solutions for these scalars can be found using equation (7.2) and (7.3):
7.2 Extrapolating Standard Distribution Models

\[ a = (N - 2((N \text{mod} 3) \text{mod} 2)) \text{div} 3 \]  \hspace{1cm} (7.2)

\[ b = (N - 3a) \text{div} 2 \]  \hspace{1cm} (7.3)

Other values are possible; however, equation (7.2) attempts to ensure that \( a > b \). This is desirable for reasons that will become clear later in the discussion.

The density \( f_{Z_N} \) of the random value representing the variable part of the payload size of the update packets \( (Z_N) \) of an \( N \)-player game can be determined using the procedure for adding independent random variables that was presented in Section 3.2.3, the result of which is that the density \( f_{Z_N} \) can be determined by \( a \) repeated convolutions of the density of the random variable representing the payload length of the 3-player game \( (f_{Z_3}) \) convolved with \( b \) repeated convolutions of the density of the random variable representing the payload length of the 2-player game \( (f_{Z_2}) \). This can be written as:

\[ f_{Z_N} = \left( \prod_{i=1}^{a} f_{Z_3} \right) \star \left( \prod_{j=1}^{b} f_{Z_2} \right) \]  \hspace{1cm} (7.4)

where \( \star \) in equation 7.4 denotes the convolution operation. As the convolution operation is computationally expensive, it is desirable to minimise the number of convolutions required for a particular value of \( N \). Therefore, the convolutions involving the density of the 3-player game should be maximised, which implies that \( (a > b) \), hence equation 7.2 is appropriate.

This technique has two disadvantages: the first is that the number of convolutions required increases with increasing \( N \), which is computationally expensive; the second is that two sets of measurements must be taken, one for a 2-player and one for a 3-player game.

The first point can be addressed by making use of the characteristic function \( \varphi_Z(\omega) \) instead of the density \( f_Z \). This simplifies Branch’s procedure as the convolution operation can be replaced by a multiplication. We showed in Section 3.2.4 that the characteristic function of the sum of random variables corresponds to the product of their respective characteristic functions. Therefore, equation 7.4 can be re-expressed as a product of characteristic functions as shown in equation 7.5.

\[ \varphi_{Z_N}(\omega) = \prod_{i=1}^{a} \varphi_{Z_3}(\omega) \times \prod_{j=1}^{b} \varphi_{Z_2}(\omega) \]  \hspace{1cm} (7.5)
As the computation of the characteristic function is expensive, the \( \varphi_{Z_2}(\omega) \) and \( \varphi_{Z_3}(\omega) \) should be computed once and stored for later use. In the case where random variates must be generated, for example, as part of a simulation, the characteristic function can be transformed back into a density. This may not be as simple as it first seems, since an inverse Fourier transform will need to be computed, (refer to equation 3.6). Alternatively, the characteristic function could be used directly to generate the random variates, using methods such as those proposed by Devroye [49]. Again, these methods may require considerable computing resources for their implementation. As an alternative, the cumulant generating function [65, p. 184] could be used instead of the characteristic function. In this case, the convolution would be transformed into a sum rather than a multiplication, which may result in further savings.

### 7.3 Extrapolation using the EG distribution

The second disadvantage of Branch and Armitage’s extrapolation technique (the need for measurements from both 2 and 3-player games) can be addressed by a simpler approach that only involves one set of measurements, for example measuring either the payload length distribution of the 2-player game or that of the 3-player game alone. To use this method a ‘per player’ density \( f_{Z_1} \) is required. Using this, the density \( f_{ZN} \) of an N-player game, can be determined from the N-fold convolution of the ‘per player’ density using the following expression:

\[
f_{ZN} = \ast_{i=1}^{N} f_{Z_1}
\]  

(7.6)

Alternatively the ‘per player’ characteristic function \( \varphi_1(\omega) \) could be used.

\[
\varphi_{ZN}(\omega) = \prod_{i=1}^{N} \varphi_{Z_1}(\omega) = (\varphi_{Z_1}(\omega))^N
\]

(7.7)

Unfortunately, this ‘per player’ distribution cannot easily be measured empirically, as it requires collecting statistics of a ‘single player’ game. Since a single player would not interact with other human players, the statistics collected from such a case would not be representative of games with larger numbers of players. However, this distribution can be inferred by a de-autoconvolution of a 2-player game by proceeding as follows.

The assumption that players behave similarly, implies that for each of the two players, the random variable describing the variable part of the update payload is
7.3 Extrapolation using the EG distribution

drawn from an identical distribution. Therefore, assuming that the random variable $Z_2$ is the sum of two iid random variables (i.e. $Z_2 = Z_{1a} + Z_{1b}$) then the density $f_{Z_2}$ can be expressed as the convolution of the ‘per player’ density $f_{Z_1}$ with itself (autoconvolution). That is:

$$f_{Z_2} = f_{Z_1} * f_{Z_1} \quad (7.8)$$

Thus, the ‘per player’ density $f_{Z_1}$ can be determined from the 2-player game by the de-autoconvolution of the 2-player game density $f_{Z_2}$. This can be computed using various approaches, for example trial and error or by using techniques such as those proposed by Choi and Lanterman [37]. These methods are challenging to implement in practice and they are computationally intensive, so avoiding the de-autoconvolution operation is desirable.

In this section, a novel approach that avoids the de-autoconvolution operation is explored. In Section 3.4.1, the Ex-Gaussian distribution was introduced and in Section 5.3.2, it was shown that this distribution is suitable for modelling the server-to-client update payload’s size of a 2-player game.

Now, assuming that the random variable $(Z_2)$ representing the variable part of the server-to-client payload of the 2-player game is in fact Ex-Gaussian distributed (equation 7.10), then its density is given by equation 7.9 (also equation 3.40), which is the sum of a Gaussian and an Exponentially distributed independent random variable. Recalling that the Exponential distribution is a special case of the Gamma distribution with $\gamma = 1$ then, as shown in Section 3.4.2, the random variable $(Z_2)$ is also Gamma Modified Gaussian (Ex-Gaussian) distributed with a characteristic function given by equation 7.11 with parameters $\mu_2, \sigma_2, \gamma_2 = 1, \theta = \frac{1}{\lambda}$.

$$f(x, \mu, \sigma, \lambda) = \lambda \exp \left( (\mu - x) \lambda + \frac{\lambda^2 \sigma^2}{2} \right) \Theta \left( \frac{x - \mu - \lambda \sigma^2}{\sigma} \right) \quad \sigma > 0 \quad (7.9)$$

$$Z_2 \sim \Gamma MG \left( \mu_2, \sigma_2, 1, \frac{1}{\lambda} \right) \quad (7.10)$$

$$\varphi_{Z_2}(\omega) = (1 - j \kappa \omega)^{-\gamma_2} \left( \exp \left( j \mu_2 \omega - \frac{\sigma^2_2 \omega^2}{2} \right) \right) \quad (7.11)$$
The $\Gamma MG$ distribution can be decomposed into two $\Gamma MG$ distributions (refer to Section 3.4.3) with the same scale parameter ($\kappa$) and with the other parameters given by:

\[
\mu_{X+Y} = \mu_X + \mu_Y, \quad \sigma^2_{X+Y} = \sigma^2_X + \sigma^2_Y \quad \gamma_{X+Y} = \gamma_X + \gamma_Y
\]  

Here $X$ and $Y$ represent the contribution from each player to the variable part of the server-to-client payload of the 2-player game. Making the same assumptions as Branch and Armitage [18] then:

\[
\mu_X = \mu_Y = \frac{\mu_2}{2}, \quad \sigma^2_X = \sigma^2_Y = \frac{\sigma^2_2}{2} \quad \gamma_X = \gamma_Y = \frac{\gamma_2}{2} = \frac{1}{2}
\]  

(7.13)

Based on the above, the ‘per player’ characteristic function can be written as:

\[
\varphi_{Z_1}(\omega) = (1 - j\kappa\omega)^{-1/2} \left( \exp \left( j\frac{\mu_2}{2} \omega - \frac{\left(\frac{\sigma^2_2}{2}\right) \omega^2}{2} \right) \right)
\]  

(7.14)

This characteristic function can be used to determine the characteristic function of the N-player game $\varphi_N(\omega)$, by taking advantage of equation 3.9 and the arguments proposed by Branch and Armitage [18]. The characteristic function of the variable part of the server-to-client payload for the N-player game is given by equation (7.7).

Substituting for $\varphi_{Z_1}(\omega)$ from equation 7.14 into equation (7.7) results in:

\[
\varphi_{Z_N}(\omega) = (1 - j\kappa\omega)^{-\frac{N}{2}} \left( \exp \left( j\frac{N\mu_2}{2} \omega - \frac{\left(\frac{N\sigma^2_2}{2}\right) \omega^2}{2} \right) \right)
\]  

(7.15)

As the $\Gamma MG$ distribution is closed under addition, then $\varphi_{Z_N}(\omega)$ must also be the characteristic function of a $\Gamma MG$ distribution, and thus the random variable $Z_N$ representing the server-to-client payload length of the N-player game also has a $\Gamma MG$ distribution, i.e.:

\[
Z_N \sim \Gamma MG \left( \frac{N\mu_2}{2}, \frac{N\sigma^2_2}{2}, \frac{N}{2}, \kappa \right)
\]  

(7.16)

Using this approach it can be shown that:

\[
Z_2 \sim \Gamma MG \left( \mu_2, \sigma^2_2, 1, \kappa \right) \sim EG \left( \mu_2, \sigma^2_2, \kappa \right)
\]  

(7.17)

\[
Z_3 \sim \Gamma MG \left( \frac{3\mu_2}{2}, \frac{3\sigma^2_2}{2}, \frac{3}{2}, \kappa \right)
\]  

(7.18)
7.4 Results from the EG extrapolation method

\[ Z_4 \sim \Gamma MG \left( 2\mu_2, 2\sigma_2^2, 2, \kappa \right) \]  

(7.19)

Where \( Z_2, Z_3, Z_4 \) are the random variables that represent the variable part of the server-to-client payload of the 2, 3 and 4-player games respectively.

Using these expressions, it can then be determined that the density of the variable part of the server-to-client payload of the N-player game is given by the convolution of a Gaussian density and a Gamma density as follows:

\[ f(z = x + g, \mu, \sigma, \gamma, \kappa) = f(x, N\mu_2, N\sigma_2^2) * f(g, N\kappa) \]  

(7.20)

The advantage of this technique, over that proposed by Branch and Armitage [18], is that it only involves a single convolution. Furthermore, this convolution may be avoided by using the explicit expression for the Gamma Modified Gaussian PDF outlined in Appendix C, or in the case of a simulation model, by exploiting the fact that this is simply the sum of Gamma and Gaussian distributed random variables.

7.4 Results from the EG extrapolation method

In this section, the new extrapolation technique outlined in Section 7.3 is compared with that proposed by Branch and Armitage [18].

The N-player distribution was synthesised by first estimating the \( \mu_2, \sigma_2^2, \kappa \) parameters of the best-fit Ex-Gaussian distribution of a 2-player game as outlined in Section 5.3.2. From these parameters, random variables distributed according to equation 7.20 were generated for different numbers of players (N). Figures 7.1 to 7.7 show the distribution plots (CDF) for the seven games for 4 and either 7 or 9-players. The CDFs show that while in general the method proposed by Branch and Armitage results in a better fit, in many cases, the difference between the two methods is not great. In some cases, Branch and Armitage’s method produces a distribution that provides a worse fit, for example the 9-player Half-Life game. Generally, the median of the distributions agrees well with that of the empirical data. The main deviations generally occur in the upper tails of the distributions, eg. Counter-Strike-Source (Figure 7.2), Quake III Arena (Figure 7.6). The fit for the Quake IV seven player game (Figure 7.7) is quite poor; however, this new technique produces results that are comparable with those from Branch and Armitage [18].

A further advantage of this new method of extrapolation is that the convolution operation is not necessary in order to generate representative payload lengths, thus
it is computationally simpler than the method of Branch and Armitage [18], which requires repeated convolution operations. The convolution is avoided since the random variables representing the payload length are generated by adding together a Gaussian distributed and a Gamma distributed random variable.

This section has outlined a new technique that can be used to synthesise representative FPS game server-to-client payload length distributions for larger numbers of players from traffic measurements from controlled game trials with two players. By making simplifying assumptions, and by making use of the properties of the EG distribution this new technique produces similar results to previously proposed techniques. This technique is useful for building realistic traffic generators for FPS game traffic simulation models, as it is computationally simpler than previous methods.
7.4 Results from the EG extrapolation method

Figure 7.3: HL2DM: Empirical and Synthetic CDFs

Figure 7.4: HLCS: Empirical and Synthetic CDFs

Figure 7.5: HLDM: Empirical and Synthetic CDFs
that require repeated convolution operations. However, the issue of the autocorrelation between successive updates has not been addressed. In the next section, this technique is applied to the time-series models for FPS games. The technique proposed in this section has been accepted for publication in [41].

7.5 Extrapolating Time-series models

We showed in Chapter 4 that the server-to-client traffic traces for the FPS games that have been used for this study mostly have autocorrelation functions that decay exponentially. This was also confirmed by Branch et al. [22]. This correlation structure needs to be incorporated into the models for FPS traffic. In this section, a generalised framework, based on the work from Chapter 6, is developed for
extrapolating the time-series behaviour of FPS game traffic so that time-series prediction models for games with a large number of players can be constructed based on measurements from data for games trials with only a small number of players.

### 7.5.1 Extrapolation of GARMA\((p,q,\delta,\eta)\) models to N-player games

The discussion of the extrapolation of the more general GARMA in this section is presented for completeness, since as was shown in chapter 6 an ARMA\((1,1)\) model is quite adequate. Section 3.11 showed that a GARMA model can be used to model a time-series that may be periodic and that may exhibit either long-range, short-range (or both) dependent effects. The GARMA\((p,q,d,\eta)\) model is defined in Section 3.11 by equation 3.80. Assuming that the time-series \(X_n\) has zero mean (i.e. only consider the variable part of the payload) then equation 3.80 can be expressed as a transfer function as follows:

\[
X_n = \frac{\Theta(B)}{\Phi(B)(1-2\eta B + B^2)^\delta} \varepsilon_n, \quad |\delta| \leq \frac{1}{2}, |\eta| \leq 1
\]  

(7.21)
or

\[
X_n = \frac{\Theta(B)}{\Phi(B)} \Upsilon^{-\delta} \varepsilon_n
\]  

(7.22)

where

\[
\Upsilon^{-\delta} = \frac{1}{(1-2\eta B + B^2)^\delta}
\]  

(7.23)

This transfer function will be used in the next section to show how the model can be extrapolated.

### 7.5.2 Repeated Convolution Extrapolation Technique

It can be shown that a GARMA\((p,q,\delta,\eta)\) process can be used to predict GARMA time-series models for N-player games, where \(N \geq 3\), by making the same simplifying assumptions as Branch and Armitage [18], repeated below:

- The nature of game play for individual players does not change significantly 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 [18].

From these assumptions, we can say that the random variable describing the behaviour of a player is independent of the number of players participating in the
N-player Extrapolation Models

game and is identically distributed (iid). Thus, the random variable describing the
behaviour of the N-players can be constructed by adding the random variables de-
scribing the behaviour of players in games with a smaller number of participants.
As previously stated, these assumptions simplify the analysis, but in doing so they
only approximate the true nature of FPS games. This proposed extrapolation pro-
cedure is similar to that used in Section 7.2, and is an extension of that published
in Branch et al. [21]. In the case where the number of players is even the procedure
is as follows:

Using equation 7.22, a 2-player game can be described by the GARMA process:

\[ X_{2,n} = \frac{\Theta(B)}{\Phi(B)} \Upsilon^{-\delta} \varepsilon_{2,n} \] (7.24)

Consider two 2-player game payload length sequences generated by the above model:

\[ X_{2,n}^{(1)} = \frac{\Theta(B)_1}{\Phi(B)_1} \Upsilon_{1}^{-\delta} \varepsilon_{2,n}^{(1)} \] (7.25)

and

\[ X_{2,n}^{(2)} = \frac{\Theta(B)_2}{\Phi(B)_2} \Upsilon_{2}^{-\delta} \varepsilon_{2,n}^{(2)} \] (7.26)

As it was assumed that player behaviour is similar, and since the residuals \( \varepsilon_{2,n}^{(1)} \)
and \( \varepsilon_{2,n}^{(2)} \) are iid, then it would be reasonable to assume that two different games
are described by the same parameters \( \Theta(B) \), \( \delta \), \( \eta \) and \( \Phi(B) \). This last assumption is
supported by the results of 6.2.2 and 6.2.5 where the parameters of the fitted ARMA
and AR models did not change significantly for various numbers of players. Based
on these assumptions, a typical 4-player game can then be described as the sum of
the sequences representing the two 2-player games:

\[ X_{4,n} = (X_{2,n}^{(1)} + X_{2,n}^{(2)}) = \frac{\Theta(B)}{\Phi(B)} \Upsilon^{-\delta} (\varepsilon_{2,n}^{(1)} + \varepsilon_{2,n}^{(2)}) \] (7.27)

this expression has the form:

\[ X_{4,n} = \frac{\Theta(B)}{\Phi(B)} \Upsilon^{-\delta} \varepsilon_{4,n} \] (7.28)

which is also a GARMA process with the same values of \( \Theta(B) \), \( \delta \), \( \eta \) and \( \Phi(B) \) as the
2-player game. The residuals of the 4-player game are given by \( \varepsilon_{4,n} = \varepsilon_{2,n}^{(1)} + \varepsilon_{2,n}^{(2)} \). If
the density of the 2-player game residuals \( \varepsilon_{2,n} \) is denoted by \( f_{\varepsilon_2} \) and that of the
4-player residuals \( \varepsilon_{4,n} \) by \( f_{\varepsilon_4} \), then the density of the 4-player game residuals \( f_{\varepsilon_4} \)
7.5 Extrapolating Time-series models

can be obtained from the convolution of \( f_{\varepsilon 2} \) with itself (auto-convolution), since it is the sum of two independent random variables (refer to Section 3.2.3). That is:

\[
f_{\varepsilon 4} = f_{\varepsilon 2} \ast f_{\varepsilon 2}
\]  

(7.29)

This process can be extended to the case where the number of players is odd by using the density of the residuals for a 3-player game. For example, the density of a 5-player game can be estimated by using:

\[
f_{\varepsilon 5} = f_{\varepsilon 3} \ast f_{\varepsilon 2}
\]  

(7.30)

Using this technique, the variable part of the payload size sequence representing the \( N \)-player game can be determined using equation 7.22 with the residuals \( \varepsilon_n \) drawn from a distribution that is formed by the required number of convolutions of the 2 and 3-player residual densities. As was stated previously, the disadvantage of this technique is that the density of both the 2 and 3-player residuals is required.

7.5.3 ‘Per player’ Residual Density

The argument presented in the previous section can also be applied to the 2-player case by recalling the assumption that players behave similarly, this implies that residuals representing the independent contribution from each of the two players is drawn from an identical distribution. Therefore, the density of the residuals of the 2-player game is given by the auto-convolution of a ‘per player’ residual density \( f_{\varepsilon 1} \). That is:

\[
f_{\varepsilon 2} = f_{\varepsilon 1} \ast f_{\varepsilon 1}
\]  

(7.31)

This ‘per player’ residual density can be used as the basis of building the density of the residuals for \( N \)-player games by repeated auto-convolutions. The \( N \)-player game can be described by:

\[
X_{N,n} = (X_{1,n}^{(1)} + X_{1,n}^{(2)} \ldots + X_{1,n}^{(N)}) = \frac{\Theta(B)}{\Phi(B)} \Upsilon^{-\delta} (\varepsilon_{1,n}^{(1)} + \varepsilon_{1,n}^{(2)} \ldots + \varepsilon_{1,n}^{(N)})
\]  

(7.32)

\[
X_{N,n} = \frac{\Theta(B)}{\Phi(B)} \Upsilon^{-\delta} \varepsilon_{N,n}
\]  

(7.33)
where \( \varepsilon_{N,n} \) is determined by the N-fold auto-convolution of the ‘per player’ residual density \( f_{\varepsilon 1} \).

\[
f_{\varepsilon N} = \left( \ast \right)^{N} f_{\varepsilon 1} \tag{7.34}
\]

Alternatively the N-fold convolution of equation (7.34) can be re-expressed as a product by making use of the characteristic function of the residuals, refer to Section 3.2.4 on page 43.

\[
\varphi_{\varepsilon N}(\omega) = \varphi_{\varepsilon 1}(\omega)^{N} \tag{7.35}
\]

The ‘per player’ residual density \( f_{\varepsilon 1} \) can be determined from the 2-player game by de-autoconvolution techniques, in a similar way that the ‘per player’ payload density of Section 7.2 was determined. The advantage of this method, over that proposed in Section 7.5.2 above, is that only one set of measurements is required as opposed to having to take measurements from 2 and 3-player games. The disadvantage is that a deconvolution must be performed. This technique is not restricted to measurements from a 2-player game, but it can be used based on measurements from games with any number of players; however, the number of deconvolutions required in order to obtain the ‘per player’ residual density becomes higher for games with more than two players, thus it becomes more difficult to obtain the ‘per player’ residual density.

### 7.5.4 ARMA Models

Although the GARMA model can be applied in general to most time-series, it has a disadvantage when one attempts to implement it as part of a simulator. The Gegenbauer polynomial (equation 3.84) must be computed; unfortunately, this expression has an infinite number of terms, and in practice, a truncated approximation must be used. If correlations, which are due to this truncation, are to be avoided, the number of elements in the truncated series must be larger than the number of samples in the trace that is being used. This becomes impractical for a simulator. A simpler model is required for a practical simulator.

The ARMA\((p,q)\) process was considered a suitable first order model (Section 6.2.5), for this case \( \delta = 0 \) in equation 7.33, so the extrapolated model for N-players is:
\[ X_{N,n} = \frac{\Theta(B)}{\Phi(B)} \varepsilon_{N,n} \quad (7.36) \]

again, the density of the residuals is determined from equation (7.34).

For the ARMA(1,1) model equation, (7.36) simplifies to:

\[ X_{N,n} = \frac{1 - \theta_1 B}{1 - \phi_1 B} \varepsilon_{N,n} \quad (7.37) \]

### 7.6 FPS ARMA extrapolation models

In Chapter 6, it was shown that the ARMA(1,1) process is a suitable model for capturing the time-series behaviour of the server-to-client payload length. Given this result, the techniques outlined in Section 7.5.4 are applied to the server-to-client packet length for the seven FPS games that are part of this study. The following sections deal with how well this model can be extrapolated using the techniques presented thus far.

The empirical results were obtained from game sessions, as outlined in Section 3.12 with the number of players ranging from 2 to 9. For Quake IV the maximum number of players was 7. The games were approximately 20 minutes in duration, with approximately 20,000 samples for each player. Only the game-play phase was included in the analysis, and where necessary any periodic components were removed from the traces.

The following method was used to evaluate the suitability of the method described above for the extrapolation from a 2-player game to an N-player game.

- Capture statistics of traffic during the active game-play phase.
- Where trends or periodic components are present in the trace, the effects of these are examined, and if necessary, these are removed to ensure that the trace data are stationary.
- Determine the time-series model parameters \((d, \phi, \theta)\) for the 2-player game.
- Determine the residuals \(\varepsilon_{2,n}\) for the 2-player game and from these determine the ‘per player’ residual density \(f_{\varepsilon_1}\) by deconvolution.
- Take N auto-convolutions of \(f_{\varepsilon_1}\) to construct the synthetic residual density of games with N players.
- Generate a sequence of packet lengths using the model and the synthetic residual density.
• Extract and plot on the same set of axes the density from the synthetic and the empirical sequence, and compare the two.

The parameters of the models are summarised in Section 6.2.5. As mentioned in Section 6.2.5 the ARMA(p,q) process is suitable as a model for the FPS games in this study as $d \approx 0$.

The next section presents the results of estimating the ‘per player’ residual density from the traces of 2-player games.

### 7.6.1 Server-to-Client Payload Residual Density

In this section the ‘per player’ residual density, as described in section 7.5.1, of the seven FPS games is presented. The residual density was obtained by the de-autocorrelation (by trial and error) of the residuals of the 2-player games, assuming an ARMA(1,1) time-series model as described in Section 7.5. The trial and error method for the de-autocorrelation was based on making an initial guess on the shape of the ‘per player’ residual density, this was convolved with itself, and the result compared with the actual density of the 2-player game residuals. The error between the computed and actual densities was used to refine the shape of the ‘per player’ residual density. The outcome of this analysis is that the residuals $\varepsilon_n$ of games with larger numbers of players can be predicted from the residuals of the 2-player game using equations 7.31 and 7.34. Figure 7.8 shows the plots of the ‘per player’ residual density for the seven games obtained by using a trial and error deconvolution technique. These plots also show the density of the empirical residuals of the 2-player game in addition to those predicted from the ‘per player’ density $f_{\varepsilon_1}$. The match is good for most of the games. The match could be improved by using a different technique to perform the de-autocorrelation, rather than trial and error, for example by the use of the techniques proposed by Choi and Lanterman [37]. However, at this stage this was not pursued, as a different strategy, similar to that of Section 7.3 that avoids the deconvolution operation, will be described in the next sections.

Figures 7.9 to 7.12 present the density of the residuals obtained by extrapolating the ‘per player’ density $f_{\varepsilon_1}$ plotted against the density of the empirically derived residuals for each game and various numbers of players. Even with the limitation of the deconvolution being performed by trial and error, it can be seen that there is an excellent match for ETPro (Figure 7.9) and HLDM (Figure 7.10), whilst for Quake4 (Figure 7.13), HLCS (Figure 7.12) and HL2DM (Figure 7.11) the match
Figure 7.8: ‘Per player’ Residual Density
is acceptable. For Quake3 (Figure 7.14), HL2CS (Figure 7.15) the match is not as good as for the other games.

The significance of these results is that the ‘per player’ residual density can be used as the basis of an extrapolation model. As this density can in principle be derived from only the measurements from a single game, it is not necessary to take measurements from games with both and even and odd number of players, as is required with the technique outlined in Section 7.5.2. Observing the densities presented in Figures 7.9 to 7.12 it seems that the shape of the residuals’ density does not vary significantly, as the number of players is increased; however, the scale parameter seems to change. This suggests that the residual’s distribution may be ‘stable’ (e.g. Gaussian, Cauchy or Lévy) or at least closed under addition. If this is indeed the case, then the parameters of the ‘per player’ residual distribution could be derived from the 2-player residual’s distribution without having to perform a deconvolution. This will be explored further in Section 7.7.

### 7.6.2 Server-to-Client Payload Length Density and ACVF

The key result presented in this section is the application of the previous analysis to predict the density and ACVF of the server-to-client payload length for the seven FPS games. Figures 7.16, to 7.22 present the results of the extrapolating the ‘per player’ residuals (Predicted) of the ARMA(1,1) model with both the empirical density (Empirical) and the density produced by using the method developed in Section 7.5.1 (Branch). Overall most of the games show a very good match between the empirical, the synthetic densities (Branch and Predicted) for the various scenarios.
7.6 FPS ARMA extrapolation models

Figure 7.10: HLDM: S2C ARMA(1,1) Residuals Density

Figure 7.11: HL2DM: S2C ARMA(1,1) Residuals Density

Figure 7.12: HLCS: S2C ARMA(1,1) Residuals Density
Figure 7.13: Quake4: S2C ARMA(1,1) Residuals Density

Figure 7.14: Quake3: S2C ARMA(1,1) Residuals Density

Figure 7.15: HL2CS: S2C ARMA(1,1) Residuals Density
The match between the ‘per player’ (Predicted) density and the empirical density for the HLDM and HLCS is excellent, even up to nine players. For the HL2DM and Quake3 games, the results of the synthetic densities (Branch and Predicted) are similar. Whilst for ETPro, HL2CS and Quake4 the ‘per player’ density method does not produce as good a match to the empirical density as does Branch’s technique. This may be because the ‘per player’ method relies on how accurately the deconvolution of the 2-player game can be done; in this case, this deconvolution was performed by trial and error. The empirical densities for HL2CS and HL2DM do not have any packets smaller than 29 and 37 bytes respectively, while Quake4 has a large spike at 27 byte packets. These peculiarities can be incorporated into a simulation model, if necessary, by truncating the synthetic densities, but at this stage the interest is focused on how well the synthetic densities can be generated from the ‘per player’ density, so these specifics have not incorporated in the modelling.

The ACVFs are determined from the ARMA(1,1) model parameters, thus there is no difference between the two extrapolation methods. The synthetic ACVFs, shown in Figures 7.23 to 7.29, generally die off more quickly than the empirical ACVFs. Higher order ARMA(p,q) models could be used if a better fit to the ACVF was required.

In most cases, the predicted results and those from Branch’s method are similar. This suggests that the server-to-client packet lengths for N-player games can be predicted from measurements from a 2-player game (or any other number of players) by the method presented in Section 7.5.4. The main disadvantage in using this method is that the density of the ‘per player’ residuals must be determined using a de-autoconvolution technique which is computationally difficult. The other disadvantage is that the density of the N-player games is determined by repeated convolutions of the ‘per player’ residual density, this is also computationally expensive. It would be advantageous if a distribution that is closed under addition could be fitted to the ‘per player’ residual density and this distribution used to determine the density of the N-player games’ residuals. In the next section, we demonstrate that the distribution of the residuals is closed under addition and that this useful fact can be used to find a computationally efficient technique for extrapolating payload distributions for FPS games.
Figure 7.16: Wolfenstein Enemy Territory Pro: Density of S2C Payload Size

Figure 7.17: Half-Life 2: Density of S2C Payload Size

Figure 7.18: Quake III Arena: Density of S2C Payload Size
7.6 FPS ARMA extrapolation models

Figure 7.19: Quake IV: Density of S2C Payload Size

Figure 7.20: Half-Life: Density of S2C Payload Size

Figure 7.21: Half-Life Counter-Strike: Density of S2C Payload Size
Figure 7.22: Counter-Strike-Source: Density of S2C Payload Size

Figure 7.23: Wolfenstein Enemy Territory Pro: ACVF of S2C Payload Size

Figure 7.24: Quake III Arena: ACVF of S2C Payload Size
7.6 FPS ARMA extrapolation models

Figure 7.25: Quake IV: ACVF of S2C Payload Size

Figure 7.26: Half-Life: ACVF of S2C Payload Size

Figure 7.27: Half-Life 2: ACVF of S2C Payload Size
7.7 ‘Per Player’ Residual Density

The extrapolation method presented in Section 7.6 provides a way of deriving representative densities of the N-player server-to-client payload size time-series from measurements taken from a 2-player game. As previously mentioned the disadvantages of this technique are that a deconvolution must be performed followed by a series of auto-convolutions. It is desirable to avoid these steps, as they are computationally intensive. In this section, a novel method is developed that allows one to predict estimates of the ‘per player’ residual density without having to resort to convolution techniques.
In Section 7.3 it was shown that by assuming that the server-to-client update packet payload size was Ex-Gaussian distributed \(X \sim \text{EG}(\theta, \mu, \sigma)\) or \(X \sim \text{GMG}(\theta, \mu, \sigma, 1)\), an extrapolation technique that avoided the auto-convolution of the ‘per player’ density was developed. The question that arises naturally from this is whether a similar technique can be used to predict the density of the residuals for the N-player time-series model.

The results of Chapter 6 showed that an ARMA(1,1) model was suitable to capture the temporal aspects of the server-to-client payload. This model can be described by equation 3.87, which is repeated here for convenience.

\[
\Phi(B)X_n = \Theta(B)\varepsilon_n
\]  

(7.38)

For the ARMA(1,1) case this becomes:

\[
X_n - \phi_1 X_{n-1} = \varepsilon_n - \theta_1 \varepsilon_{n-1}
\]  

(7.39)

From this expression, it can be seen that the random variable denoting the current residual \((\varepsilon_n)\) minus a scaled contribution of the random variable denoting the past residual \((\theta_1 \varepsilon_{n-1})\) is equal to a term which is related to the difference between the random variable denoting the current sample and a scaled version of the random variable denoting the previous sample \((X_n - \phi_1 X_{n-1})\). This implies that the density of the random variable denoting the current residual \(f_{\varepsilon}\) can be determined from the density of the random variable denoting the previous residual \(f_{\varepsilon_{n-1}}\) and the density of the random variable denoting the \((X_n - \phi_1 X_{n-1})\) term. As the residuals are iid the densities \(f_{\varepsilon_n}\) and \(f_{\varepsilon_{n-1}}\) are the same. In the case of the random variables denoting the current and previous samples \(X_n\) and \(X_{n-1}\) these are also identically distributed, however, they are correlated so they are not independent. The sum of correlated random variables is not given by equation 3.4, but by equation 7.40 [101]:

\[
f_Z(z) = \int_{-\infty}^{\infty} f_{XY}(z - x)dx
\]  

(7.40)

This correlation makes it difficult to obtain the density of the \((X_n - \phi_1 X_{n-1})\) random variable as the joint density must be known. It is difficult to determine an expression for the joint density without making assumptions about the distribution of the random variables involved. For example, that the random variables are Gaussian distributed.
7 N-player Extrapolation Models

To simplify the analysis, one could assume that the joint density is small and thus an approximation to the true density of the residuals could be found. However, it must be stressed that the expressions for the density of the residuals in these circumstances would be approximate and not exact, since in fact the joint density may be large and cannot be ignored.

In the case of the 2-player game, the packet payload lengths are assumed to be EG distributed. As previously noted this distribution is the $\Gamma_{MG}$ distribution with a shape parameter equal to one. It is more convenient to work with the characteristic function rather than the density. Using the results of Section 3.4.3, the characteristic function of the random variable denoting the $(X_n - \phi X_{n-1})$ term for the 2-player game is given by:

$$
\varphi_{X-\phi X}(\omega) = \left(1 + \phi^2 \kappa^2 \omega^2 - j\kappa (1 - \phi) \omega\right)^{-1} \exp\left[j (1 - \phi) \mu_2 \omega - \frac{(1 + \phi^2) \sigma_2^2 \omega^2}{2}\right]
$$

This is a $V_{\Gamma_{MG}}$ distribution as defined in Section 3.4.3. The above expression assumes that $X_n, X_{n-1}$ are independent which is not the case.

The residuals, on the other hand, are iid, so if the assumption that the $X_n, X_{n-1}$ are independent was true, then the difference of the residuals $\varepsilon_n - \theta_1 \varepsilon_{n-1}$ will also be $V_{\Gamma_{MG}}$ distributed. In this case the characteristic function is:

$$
\varphi_{\varepsilon_n}(\omega) \varphi_{\varepsilon_{n-1}}(-\theta \omega) = (1 - j\kappa \omega)^{-1} \exp\left[j \mu_2 \omega - \frac{\sigma_2^2 \omega^2}{2}\right] (1 + j\theta \kappa \omega)^{-1} \exp\left[-j \theta \mu_2 \omega - \frac{\sigma_2^2 (-\theta \omega)^2}{2}\right]
$$

This is the product of the characteristic functions of two $\Gamma_{MG}$ distributions. From this analysis, we conclude that the residuals are $\Gamma_{MG}$ distributed with:

$$
\varphi_{\varepsilon_n}(\omega) = (1 - j\kappa \omega)^{-1} \exp\left[j \mu_2 \omega - \frac{\sigma_2^2 \omega^2}{2}\right]
$$

$$
\varphi_{\varepsilon_{n-1}}(-\theta \omega) = (1 + j\theta \kappa \omega)^{-1} \exp\left[-j \theta \mu_2 \omega - \frac{\sigma_2^2 (\theta \omega)^2}{2}\right]
$$

Unfortunately, using this approach does not yield results that are similar to those obtained from the empirical traces. This is probably due to the incorrect assumption that the joint density of $X_n$ and $X_{n-1}$ is small and that the $X_n, X_{n-1}$ samples are independent.
7.7 ‘Per Player’ Residual Density

Consequently, a different approach is used to find a distribution that better matches the empirical residuals. The assumption here being that the residuals are VTG distributed, that is

\[ \varepsilon_n \sim (1 + \theta_1) VTG(\mu, \sigma, \gamma, \kappa, \phi_1) \] (7.45)

As this distribution is also closed under addition (refer to Section 3.4.3) the ‘per player’ residuals are VTG distributed and so are the residuals of the N-player games. This simplified analysis leads to a simple method of estimating the ‘per player’ residual density from the density of the 2-player server-to-client update packet payload size density and the \( \phi_1, \theta_1 \) parameters of the ARMA(1,1) time-series model.

\[ \varepsilon_{1,n} \sim (1 + \theta_1) VTG \left( \frac{\mu_2}{2}, \frac{\sigma_2^2}{2}, \frac{1}{2}, \kappa, \phi_1 \right) \] (7.46)

Similarly, the residuals of the N-player game can be determined by using:

\[ \varepsilon_{N,n} \sim (1 + \theta_1) VTG \left( \frac{N\mu_2}{2}, \frac{N\sigma_2^2}{2}, \frac{N}{2}, \kappa, \phi_1 \right) \] (7.47)

where \( \mu_2, \sigma_2^2, \kappa \) are the best-fit EG parameters of the 2-player game.

The following method was used to evaluate the suitability of the technique described above for the extrapolation from a 2-player game to an N-player game.

- Capture statistics of traffic during the active game-play phase.
- Where trends or periodic components are present in the trace, the effects of these are examined, and if necessary, these are removed to ensure that the trace data are stationary.
- Determine the time-series model parameters(\( \phi, \theta \)) for the 2-player game.
- Determine the parameters of the best fit Ex-Gaussian for the server-to-client packet payload of the 2-player game. From these determine the parameters for the ‘per player’ residual density.
- Use equation 7.47 to construct synthetic residual density of games with N players.
- Generate a sequence of packet lengths using the model and the synthetic residual density.
- Extract and plot on the same set of axes the density of the synthetic and the empirical sequence for comparison.

The next section presents the results of using this method.
Table 7.1: Timings: Repeated Convolutions vs VΓ technique

<table>
<thead>
<tr>
<th>Convolution size</th>
<th>Time to calculate convolution (ms)</th>
<th>Time to generate residuals (ms)</th>
<th>VΓ Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>7.59</td>
<td>30.08</td>
<td>4.63</td>
</tr>
<tr>
<td>1024</td>
<td>8.31</td>
<td>57.65</td>
<td>4.67</td>
</tr>
<tr>
<td>2048</td>
<td>9.50</td>
<td>110.80</td>
<td>4.55</td>
</tr>
</tbody>
</table>

7.8 Extrapolating the Residual Density

7.8.1 The Variance Gamma Modified Gaussian Density as the Residual Density

In this section, the VΓ distribution is used as the ‘per player’ residual distribution. This distribution is used as a simple means of extrapolating the ‘per player’ residuals to games with N players.

Figures 7.30 to 7.36 present the results from extrapolating the residuals of the ARMA(1,1) model using the technique outlined in Section 7.7 using the parameters from the best-fit Ex-Gaussian distribution of the 2-player games for the various FPS games. The results (Synthetic) are plotted against the empirically obtained results and those obtained using the repeated convolution technique of Section 7.5. As can be seen the results obtained by this extrapolation technique compare very well with those using the repeated convolution method.

The Quantile-Quantile Plots show that the deviations occur mainly in the upper tails of the distributions. This is consistent for both extrapolation techniques. The match between the empirical and extrapolated results is compared to a maximum number of nine players. The results of this study are positive. A number of the games have a minimum payload size that is larger than that predicted by the extrapolation technique, for example: Counter-Strike-Source, Half-Life 2, Half-Life Counter-Strike, and Half-Life. This minimum payload size can be incorporated in the fixed part of the packet, as this analysis is based around the variable part of the payload.

This new technique is also computationally simpler than the repeated convolution method. Indicative run times to generate 10000 samples of the residuals for a 16 player game are given in Table 7.1. These results were obtained using the MATLAB ‘tic’ and ‘toc’ functions. As can be seen the new technique is considerably faster.

In summary, the new extrapolation technique outlined in Section 7.7 provides estimates of the packet payload size that are not dissimilar (in some cases better) to those obtained by using an N-fold convolution technique, without the computational burden of this last technique. However, care must be exercised when using this
7.8 Extrapolating the Residual Density

[Graphs and diagrams are shown here but not transcribed.]

Figure 7.30: Wolfenstein Enemy Territory Pro: S2C Packet Payload Size: ARMA(1,1) Extrapolated

approach (or previous ones for that matter), as the synthesised payload’s density has a support that may include negative numbers (for example as is the case for the 9-player Half-Life game). To avoid this situation the densities can be truncated to a particular minimum value. As discussed previously, the extrapolation techniques are based around the variable part of the server-to-client payload size. The fixed part of this payload has been simply modelled by an offset parameter that is extrapolated by multiplying it by the number of players participating in a game. Estimating this offset parameter from the from the 2-player game trace alone is not reliable. As can be seen from Figures 7.30 to 7.36 the median of the synthetic distributions does not always match that of the empirical distributions. For this work, this parameter has been estimated from the 2 and 4-player traces in order to obtain a reasonable estimate. However, a more reliable method of estimating this parameter is required; this is an area for further research.
7.9 Conclusion

In this chapter, a new technique has been outlined that can be used to synthesise representative FPS game server-to-client payload length distributions for larger numbers of players based on measurements from controlled trials with two players. By making use of the properties of the Ex-Gaussian, distribution this new technique, which is based on a ‘per player’ density, produces similar results to previously proposed extrapolation techniques. This technique is useful for building realistic traffic generators for FPS game traffic simulation models, as it is computationally simpler than previous methods which require repeated convolution operations.
This Chapter has also outlined novel techniques for extrapolating the ARMA(1,1) models of 2-player FPS games to games with larger numbers of players by using the properties of the $\Gamma$MG and VTG distribution to predict the density of the residuals of games with $N$ players from the residuals of the 2-player game. This technique produces results that match the empirical distributions well,

This technique should facilitate the construction of FPS game traffic simulators that allow small-scale empirical measurements to be applied to larger scale simulations of FPS traffic in networks. This topic will be discussed further in the next chapter.
Figure 7.33: Half-Life Counter-Strike: S2C Packet Payload Size: ARMA(1,1) Extrapolated
Figure 7.34: Half-Life: S2C Packet Payload Size: ARMA(1,1) Extrapolated
Figure 7.35: Quake III Arena: S2C Packet Payload Size: ARMA(1,1) Extrapolated
Figure 7.36: Quake IV: S2C Packet Payload Size: ARMA(1,1) Extrapolated
Chapter 8

Simulation Models

8.1 Introduction

This chapter deals with the implementation of a traffic generator based on the model developed in Chapter 7 and the INET Framework of OMNeT++.

The chapter is organized as follows. Section 8.2 presents a description of the traffic generator based on the findings of Chapter 7. Section 8.3 presents validation of the simulation traffic model against the results from Chapter 7. Section 8.4 presents some results comparing of the model of Chapter 7 and that proposed by the IEEE 802.16 Broadband Wireless Access Working Group [129], while in Section 8.5 conclusions are presented.

8.2 OMNeT++ based First-person Shooter (FPS) Game Traffic Generator

The model of Section 7.7 was implemented as a traffic generator using the INET Framework [135] of OMNeT++ [136]. The INET Framework is an open-source data networking simulation package for the OMNeT++ simulation environment[135]. This framework contains simulator implementations of various communications protocols together with implementations of various networking nodes and several application models. The INET framework uses a layered approach that mimics the TCP/IP network stack. In the INET Framework, protocols and applications are represented by modules whose external interface is defined in a ‘.ned’ file. These simulation modules communicate with each other by generating and passing messages between themselves. Each module is implemented as a C++ class with the
same name as the `.ned` file. These modules can be combined to form hosts and other network devices by using the Network Description (NED) language [135]. The traffic generator models in this study were developed using OMNeT++ version 4.0 and INET Framework version INET-20100723.

### 8.2.1 General Traffic Generator Details

The traffic generator produces messages (UDP packets) whose payload length and inter-packet times are either drawn from distributions specified by probability density functions or by techniques based on the models of Chapter 7. Since the simulator may support different game traffic generators, a base module that can be used to develop code for specific game traffic models is provided (GameBasicApp) (for details of each module the reader is referred to Appendix B). This base module contains the basic parameters, statistics collection, and the interface for the traffic generator application. This module consists of two parts. A `.ned` file (GameBasicApp.ned) which specifies the external interface and the simulation parameters that are required by the simulation model, and the corresponding file that contains the C++ code that implements the generator (GameBasicApp.cc). The traffic generator application is a C++ class that is a child of the INET framework’s `UDPAppBase` class. This last class contains the required methods that handle the sending and receiving of UDP packets. The GameBasicApp module is included as an application in the INET framework ‘applications’ directory. The details of the methods, parameters and files are described in Appendix B.1.1

The ARMA traffic simulator is based on the work outlined in Section 7.7, and is implemented in the ‘ExGaussArmaApp.ned’, and ‘ExGaussArmaApp.cc’ files. The ‘ExGaussArmaApp.ned’ file specifies the interface into the traffic generator application, while the ‘ExGaussArmaApp.cc’ contains the traffic generator (C++) code for the methods that have been redefined from the ‘GameBasicApp’ class. The main implementation of the model resides in the ‘ExGaussArmaApp.cc’ file (see Appendix B.1.2 and B.7).

### 8.3 Traffic Generator Validation

The traffic generator was tested with different numbers of clients to ensure that the payload lengths and the inter-packet times were consistent with those of Section 4.2. The client-to-server Inter-arrival times (IATs) are consistent with the results
### Table 8.1: Distribution Parameters for S2C Packet Payload Length

<table>
<thead>
<tr>
<th>Game</th>
<th>Players</th>
<th>Payload C2S</th>
<th>Payload S2C</th>
<th>IAT C2S</th>
<th>IAT S2C</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>std</td>
<td>mean</td>
<td>std</td>
<td>mean</td>
<td>std</td>
</tr>
<tr>
<td>ETPro</td>
<td>2</td>
<td>31.83</td>
<td>2.80</td>
<td>85.10</td>
<td>24.15</td>
<td>21.96</td>
</tr>
<tr>
<td>HL2CS</td>
<td>2</td>
<td>30.72</td>
<td>3.95</td>
<td>65.51</td>
<td>20.28</td>
<td>62.50</td>
</tr>
<tr>
<td>HL2DM</td>
<td>2</td>
<td>55.35</td>
<td>10.84</td>
<td>110.18</td>
<td>43.81</td>
<td>50.07</td>
</tr>
<tr>
<td>HLCS</td>
<td>2</td>
<td>42.68</td>
<td>7.06</td>
<td>65.90</td>
<td>41.15</td>
<td>6.35</td>
</tr>
<tr>
<td>HLDM</td>
<td>2</td>
<td>46.57</td>
<td>6.31</td>
<td>71.37</td>
<td>35.85</td>
<td>8.08</td>
</tr>
<tr>
<td>Quake3</td>
<td>2</td>
<td>32.38</td>
<td>3.27</td>
<td>74.52</td>
<td>14.59</td>
<td>10.15</td>
</tr>
<tr>
<td>Quake4</td>
<td>2</td>
<td>51.49</td>
<td>10.36</td>
<td>103.54</td>
<td>38.36</td>
<td>29.74</td>
</tr>
<tr>
<td>IEEE 802.16</td>
<td>2</td>
<td>41.78</td>
<td>7.28</td>
<td>10.81</td>
<td>7.37</td>
<td>44.46</td>
</tr>
</tbody>
</table>

From Table 4.1, there are differences in the server-to-client IATs for Wolfenstein Enemy Territory Pro (ETPro), Counter-Strike-Source (HL2CS) and Half-Life Counter-Strike (HLCS); these differences are due to periodic components that have been included in the simulation model. The density of the simulated server-to-client payload lengths (OMNet++ Simulation) are plotted against the empirical data and the results from Section 7.8 (Gamma Modified Gaussian (ΓMG) Extrapolation) (assuming that residuals are Variance Gamma Modified Gaussian (ΓVG) distributed), refer to Figure 8.1. The simulated payload lengths produced by the traffic generator are consistent with those obtained in Section 7.8. In the case of the Quake IV (Quake4) simulation, a significant number of periodic packets of 25 byte payload were produced by the traffic generator, hence the difference between the simulated and ΓMG CDFs. (The large standard deviations in the IATs are due to the periodic packets.)

Färber [60], Park et al. [102] and Lang et al. [88] presented simple extrapolation models for HLCS and Quake III Arena (Quake3), details of these are provided in Section 2.7.6. The predicted mean server-to-client payload length for varying numbers of players produced by this model is compared with those from Färber [60], Park et al. [102] and Lang et al. [88] (Figure 8.2). These results are pleasing, especially for Quake3 where the traffic generator produced similar results to those in the literature. For HLCS, on average the traffic generator produces larger payloads that what is predicted using technique presented in Färber [60].

In the next section, the traffic generator is used in a simulation to compare it with the FPS traffic model presented in the IEEE 802.16m Evaluation Methodology Document (EMD) [129].
Figure 8.1: CDF S2C Mean Packet Payload Size
8.4 IEEE802.16 Model vs \( \Gamma \)MG Model

The simulation scenario of Figure 8.3 was used in a simple simulation of FPS game traffic (for the different game models) mixing with the traffic from an FTP file transfer. The simulation scenario shown in Figure 8.3 mimics a ‘LAN party’ with the game server on the Internet, whilst another user is downloading a large file. The \textit{FTPSource} and \textit{FTPSink} are INET framework ‘standard hosts’, the source is configured to begin transferring a 15 MB file to the sink after 2 seconds from the start of the simulation. The various link details are shown in Table 8.2.

Figure 8.4 shows the mean throughput (packets per seconds) measured at the FTP Source for the various games models and differing numbers of players. From these results, we see that increasing the number of clients results in an increase in the time taken for the FTP source to complete the file transfer. The scenarios, when the Quake models are used, result in a significantly longer time for the file to be transferred and a lower throughput. Of interest is also the fact that some games are similar to the IEEE802.16 model for the 2-player scenario, but these produce significantly different results as the number of clients is changed. As expected, the results for IEEE802.16 model do not vary significantly, as the mean server-to-client payload length remains constant irrespective of the number of players in the game. In this case, the variation of the results is due to the increased numbers of packets that flow due to the increased numbers of players.
Table 8.2: Simulation Scenario Parameters

<table>
<thead>
<tr>
<th>Link</th>
<th>Type</th>
<th>Speed</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client to Home modem</td>
<td>Fast Ethernet</td>
<td>100 Mbps</td>
<td></td>
</tr>
<tr>
<td>Home modem to ISP</td>
<td>ADSLup</td>
<td>1 Mbps</td>
<td>1 ms</td>
</tr>
<tr>
<td>ISP to Home modem</td>
<td>ADSLdown</td>
<td>12 Mbps</td>
<td>1 ms</td>
</tr>
<tr>
<td>ISP to Internet</td>
<td></td>
<td>512 Mbps</td>
<td>10 ms</td>
</tr>
<tr>
<td>Internet to Game Server</td>
<td></td>
<td>512 Mbps</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

Figure 8.3: OMNeT++ Simulation Scenario

8.5 Summary

In this chapter, a description of the implementation of a simple traffic generator that produces traffic based on the model developed in Chapter 7 and the INET Framework of OMNeT++ is provided. The traffic generator presented in this chapter can be used to investigate network performance issues when FPS games and other applications interact in an IP based network. The performance of this Vtg residual distribution based traffic generator has been compared with that proposed by the IEEE 802.16 Broadband Wireless Access Working Group [129] and it has been shown that it is superior as it can be scaled to model the effects of differing numbers of players participating in the game. It was also demonstrated that there is a significant difference between the two models. The proposed model incorporates both correlation between successive packets and the variation in payload length with differing numbers of players, while the IEEE 802.16 does not. The effects that have been modelled in this thesis, should be taken into account in any simulation involving FPS game traffic.
Figure 8.4: Mean Throughput
Chapter 9

Conclusions

9.1 Summary

The purpose of this study is to explore the nature of First-person Shooter (FPS) traffic in order to determine which characteristics of this class of traffic are necessary for building a simple model that captures these characteristics. The increased popularity of on-line multi-player games has made this work important as the models can be used by both Internet Service Providers (ISPs) and network design engineers for planning and provisioning network infrastructure. The importance of developing a model for FPS traffic has also been acknowledged by the IEEE 802.16 Broadband Wireless Access Working Group TGm [129]. Although a large body of knowledge in the area of modelling the traffic produced by on-line games exists, the models for FPS game traffic, in the game-play phase, are still relatively unsophisticated and leave many questions unanswered relating to issues around which traffic parameters are important when characterising this type of traffic.

This thesis has addressed many of these questions:

1. Which parameters of FPS traffic are ‘interesting’ in terms of formulating a model of FPS traffic?
2. What types of statistical distributions are suitable for modelling FPS traffic?
3. Can a simple model capture both the statistical nature of the size of the packets and the temporal characteristics of the traffic?
4. If a simple model can be found, can it be extrapolated so that representative traffic for games with large numbers of participants is determined from measurements of games with a small number of participants?
5. Can a computationally simple technique be used for the extrapolation?
To determine which characteristics of the traffic during the game-play phase were of interest in modelling FPS traffic, the packet traces from seven popular games were analysed. The difference between this work and previous work is that traffic from games that were run under controlled documented conditions was used, rather than traffic obtained from game servers that were hosted on the Internet. The results of this analysis showed that the server-to-client inter-arrival times were mostly deterministic, and the client-to-server Inter-arrival times (IATs), although having some spread, it was not significant, as typically the variance was small. The analysis also confirmed that the distributions of the packet payload lengths are of more interest as these showed more variability than did the inter-arrival times. For the games studied, the mean payload length in the server-to-client direction was proportional to the number of players participating in the game, whilst in the client-to-server direction the mean payload size did not depend on the number of players participating in the game. These results were not surprising as they confirmed similar findings of previous work. As some authors suggested that there are correlation effects in FPS traffic, an analysis of the nature of the correlation was conducted and it was found that the Autocovariance Function (ACVF) typically died off exponentially in the server-to-client direction. Analysis of the ACVF indicated that there was a stronger relationship between the payload size of the current packet size and the previous ones in the server-to-client direction than in the client-to-server direction. This analysis also revealed an interesting feature of the traffic, which has been overlooked in the past, that is, that some games have significant periodic traffic flowing between the client and server. These effects should be included in the models for this class of traffic.

To answer the second question, three commonly used distributions for FPS traffic modelling, namely the Gaussian, Log-Normal, Generalised Extreme Value and a mixture distribution, the Ex-Gaussian were fitted to the payload traffic traces for two-player games. The results of this part of the study revealed that in the client-to-server direction the update payload size is suitably modelled, after removing the ‘spikes’ due to periodic traffic, by one or more of these four distributions. This conclusion was based on the fact that there was little difference in the client-to-server $\chi^2$ discrepancy measure metric for the fit of the four distributions and because of the small variation in the payload size distribution. On the other hand, with the exception of the Quake IV game trace, the Generalised Extreme Value and Ex-Gaussian distributions were found to produce superior fits to the distribution of the server-to-client payload empirical data. Both these distributions are versatile as their shape
can be adjusted to fit the data. Although one finds that the Extreme Value distribution has been the most common distribution chosen to model FPS packet size distributions, the results of this work have shown that another skewed distribution (Ex-Gaussian) also models the observed data very well. This result is extremely useful, as the Ex-Gaussian distribution has the property of being closed under addition. This last property is advantageous in overcoming one of the shortcomings of the modelling techniques in the literature that rely on fitting distributions to each of the game scenarios to be modelled. By using the Ex-Gaussian distribution, a method of scaling the models is possible without having to conduct and collect data from a large number of controlled experiments.

The second shortcoming of the models described in the literature relates to our third question. Most of the models in the literature do not incorporate any serial correlation that may exist in the packet traffic. In other words, these models assume that the packets’ payload sizes are independent of each other. Having shown that this was not the case for server-to-client traffic, a simple method incorporating this correlation was developed based on time-series techniques. The GARMA time-series model was chosen as a starting point so that a general modelling framework was established. On further investigation, it was found that this last model, although general, was unnecessarily complicated for the traffic being studied, and that simpler models could be used. Despite claims by others, we found that there was no strong evidence that the traffic had Long-Range Dependent (LRD) effects. Consequently, the FARIMA models were not considered and the focus of the modelling turned to the Short-Range Dependent (SRD) models. As the goal of this work was to choose the simplest model possible, initially an AR(1) time-series model was investigated as a way to describe the temporal characteristics of the server-to-client payload size. It was shown that the AR(1) model was only moderately successful in capturing the variance of the packet lengths and that higher order AR(p) models led to a better match between the statistics of the empirical and synthetic traces in many cases. However, higher order models lack parsimony and have a consequent risk of the model being over-fitted to a particular trace. The ARMA(p,q) time-series models were also evaluated. The simplest of these (ARMA(1,1)) produced synthetic traces whose variance was better matched to the variance obtained from the AR(1) process. Again, increasing the model order generally led to a better fit as measured by the Akaike Information Criterion. However, in many cases the fit for the variance was not improved significantly, thus it was found that the ARMA(1,1) model was adequate in this respect.
Having concluded that a time-series model was appropriate in order to include the autocorrelation of the update packet size, the fourth and fifth questions were addressed; that is, determining a means of extrapolating the model so that representative traffic for games with large numbers of participants could be determined from measurements of games with a small number of participants. As the client-to-server payload length did not vary significantly with varying numbers of players only the server-to-client traffic was considered. Although some initial work had been done in this area, the techniques developed were either specific to a particular game, not general, or relied on repeated convolutions in order to extrapolate the model. We found that the Ex-Gaussian distribution provides a good fit to the server-to-client payload traffic produced by the 2-player games. Using this knowledge, and taking advantage of the properties of the Ex-Gaussian distribution, as well as making the same simplifying assumptions that previous researchers have made, has led to new techniques that can be used to synthesise representative FPS game server-to-client payload length distributions for larger numbers of players from traffic measurements from controlled game trials with two players. The techniques that have been developed avoid repeated convolution by exploiting the characteristics of the Gamma Modified Gaussian distribution, of which the Ex-Gaussian distribution is a special case. Using this distribution, the concept of a ‘per player’ distribution was developed and this was used as the basis for a simple extrapolation model. This technique was then extended in order to determine the ‘per player’ residuals of the time-series model distribution from the distribution of server-to-client payload size and thus an extrapolation technique for the time-series models was also developed. The importance of these techniques is that convolution, or deconvolution for that matter, is avoided making the extrapolation model computationally simple.

These contributions have been consolidated in a traffic model that has been implemented in a software based simulation system that has demonstrated divergence from previous models. This divergence is due to the new traffic model incorporating both the variations due to differing numbers of players, and the autocorrelated nature of the traffic.

9.2 Contributions

The work in this thesis has led to the development of a computationally simple model for First-person Shooter game traffic. A summary of the main contributions
9.3 Future research

is as follows:

- Proposed that the Ex-Gaussian distribution is a suitable model for the payload size of the update packets flowing in the server-to-client direction for a two-player game. Previous work used either the Extreme Value or Log-Normal distributions. Unlike the other distributions that have been used as FPS models; the Ex-Gaussian can be decomposed into a distribution that is closed under addition in certain circumstances. This property can be exploited to form the basis of a simple extrapolation technique.

- Proposed that the correlations in the size of update packets flowing in the server-to-client direction be modelled using a time-series approach. We demonstrated that an ARMA(1,1) model is a suitable model. Previously, the correlation was not included or very limited Markov models were used.

- Proposed a computationally simple technique that can be used to produce representative distributions of the size of the update packets flowing in the server-to-client direction from measurements of a 2-player game. Previous models either required repeated convolution operations of the models for 2-player and 3-player games or were specific to particular games.

- Proposed a technique that can be used to determine the ‘per player’ residuals of an ARMA(1,1) time-series model distribution from the distribution of the server-to-client payload size and thus an extrapolation technique that can be used for the time-series models was also developed.

- Developed a traffic generator written for Omnet++ based on the models developed in this work. This FPS traffic generator allows small-scale empirical measurements to be applied to larger scale simulations of this type of traffic in networks.

9.3 Future research

This study has developed a model for FPS game traffic culminating in its implementation as a traffic generator for use with the OMnet++ simulator. In developing this model, several simplifying assumptions were made. Whilst making these assumptions did not affect the overall results, these should be challenged in order to confirm that these assumptions are in fact valid. One of the basic assumptions was that the random variables associated with each player’s contributions were independent and thus these could be summed to produce the random variable representing the server’s
update packet size. A consequence of not making this assumption is that a more complex mathematical scenario would arise, however, it could possibly lead to a simpler model. This issue should be examined further. The current model is based on taking measurements from a 2-player game and using these measurements to determine the parameters of the best fit Ex-Gaussian distribution and the eventual estimation of the ‘per player’ residual distribution. This approach has the limitation that measurements from a 2-player game must be available. If the Gamma Modified Gaussian distribution, rather than the Ex-Gaussian distribution, is used as the basis of the model, then the statistics taken from games with arbitrary numbers of players can be used to determine the ‘per player’ distributions. In order to achieve this, a Maximum Likelihood Estimation (MLE) technique must be developed so that the parameters of the best fit Gamma Modified Gaussian distribution can be estimated from the empirical data. Developing a MLE technique requires that an expression for the likelihood function of the Gamma Modified Gaussian be determined. Once the parameters of the Gamma Modified Gaussian distribution have been determined, then the parameters of the ‘per player’ distribution can be calculated.

Another limitation of the model is that a better method to estimate the fixed part of the server-to-client update packet size is required. Currently, two measurements are needed to obtain a good estimate of the fixed part of the update packet. It is desirable to be able to estimate this with only one measurement.

The empirical traces for games used in this study were obtained from games with up to nine players. Having data from controlled game trials with a larger numbers of players would be desirable, as the accuracy of the extrapolation can be better assessed when packet fragmentation occurs on a large scale. This may have effects on the correlation nature of the traffic.

The ACVF for the server-to-client packet using the ARMA(1,1) model dies off more quickly than the empirical ACVF for some games. Higher order models could be used. However, this would lead to a more complicated extrapolation model. It is possible that a mixture of AR models could be used as an alternative.

The client-to-server traffic has only been modelled using standard distributions in this work, as the statistics of its payload size did not vary significantly and these were mostly independent of the number of players participating in the game. Questions remain such as, “Is there a correlation between the payload size of client-to-server packets and can knowledge of their distribution be used to determine the distribution of the server-to-client payload?” “How does the player behaviour or
the nature of the game and the game scenario (type of map, the player ability, the number of ‘kills’ per second) affect the parameters of the model?"

Summing up, areas for future research will involve investigation into the finding the limits of the assumptions that have been made in generating the traffic models developed in this work, and into how these models should be modified for newer FPS games.

FPS games, although they place heavy demands on network resources, they are not the only popular genre of on-line games. Future research will involve the investigation of the application of the techniques described in this work to other game genres.

This thesis has investigated an increasingly important class of traffic and proposed traffic models that match the observed traffic well, and are computationally simpler than those proposed in the past. However, the eventual purpose of developing traffic models is to use them to investigate performance issues in communications networks. The models developed in this work will aid in the investigation of how FPS games interact with other applications, and which techniques are likely to be successful in minimizing the impact on each other's traffic. Ultimately, this is the area of further research to which the work in this thesis has contributed.
References


REFERENCES


REFERENCES

workshop on Network and system support for games, NetGames ’06, New York, NY, USA. ACM.


# Appendix A

## Glossary

List of acronyms used in the text.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACF</td>
<td>Autocorrelation Function</td>
<td>36</td>
</tr>
<tr>
<td>ACVF</td>
<td>Autocovariance Function</td>
<td>198</td>
</tr>
<tr>
<td>AIC</td>
<td>Akaike Information Criterion</td>
<td>62</td>
</tr>
<tr>
<td>ARIMA</td>
<td>Autoregressive Integrated Moving Average</td>
<td>37</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive Moving Average</td>
<td>37</td>
</tr>
<tr>
<td>BFW</td>
<td>Broadband Fixed Wireless</td>
<td>16</td>
</tr>
<tr>
<td>BITSS</td>
<td>Broadband Internet Traffic Simulation and Synthesis</td>
<td>4</td>
</tr>
<tr>
<td>C2S</td>
<td>Client-to-Server</td>
<td>23</td>
</tr>
<tr>
<td>c2s</td>
<td>client-to-server</td>
<td></td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Distribution Function</td>
<td>24</td>
</tr>
<tr>
<td>CoD2</td>
<td>Call of Duty 2 version 1.0</td>
<td>38</td>
</tr>
<tr>
<td>CV</td>
<td>Coefficient of variation</td>
<td>76</td>
</tr>
<tr>
<td>EG</td>
<td>Ex-Gaussian</td>
<td>51</td>
</tr>
<tr>
<td>EMD</td>
<td>Evaluation Methodology Document</td>
<td>224</td>
</tr>
<tr>
<td>ETPro</td>
<td>Wolfenstein Enemy Territory Pro</td>
<td>191</td>
</tr>
<tr>
<td>EV</td>
<td>Extreme Value</td>
<td>39</td>
</tr>
<tr>
<td>FARIMA</td>
<td>Fractional Autoregressive Integrated Moving Average</td>
<td>10</td>
</tr>
<tr>
<td>FPS</td>
<td>First-person Shooter</td>
<td>197</td>
</tr>
<tr>
<td>fps</td>
<td>first-person shooter</td>
<td></td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
<td>220</td>
</tr>
<tr>
<td>GARMA</td>
<td>Generalized Autoregressive Moving Average</td>
<td>38</td>
</tr>
<tr>
<td>GEV</td>
<td>Generalised Extreme Value</td>
<td>48</td>
</tr>
<tr>
<td>ΓMG</td>
<td>Gamma Modified Gaussian</td>
<td>229</td>
</tr>
<tr>
<td>H</td>
<td>Hurst parameter</td>
<td>38</td>
</tr>
<tr>
<td>HLDM</td>
<td>Half-Life</td>
<td>67</td>
</tr>
<tr>
<td>HL2DM</td>
<td>Half-Life 2</td>
<td>67</td>
</tr>
<tr>
<td>HLCS</td>
<td>Half-Life Counter-Strike</td>
<td>191</td>
</tr>
<tr>
<td>HL2CS</td>
<td>Counter-Strike-Source</td>
<td>191</td>
</tr>
<tr>
<td>IAT</td>
<td>Inter-arrival time</td>
<td>198</td>
</tr>
<tr>
<td>iat</td>
<td>inter-arrival time</td>
<td>198</td>
</tr>
<tr>
<td>iid</td>
<td>independent and identically distributed</td>
<td>53</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
<td>2</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
<td>197</td>
</tr>
<tr>
<td>K-S</td>
<td>Kolmogorov-Smirnov</td>
<td>25</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
<td>24</td>
</tr>
<tr>
<td>LogN</td>
<td>Log-Normal</td>
<td>39</td>
</tr>
<tr>
<td>LRD</td>
<td>Long-Range Dependent</td>
<td>199</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimation</td>
<td>202</td>
</tr>
<tr>
<td>MM</td>
<td>Method of Moments</td>
<td>55</td>
</tr>
<tr>
<td>MMOG</td>
<td>Massively Multiplayer On-line Games</td>
<td>2</td>
</tr>
<tr>
<td>PACF</td>
<td>Partial Autocorrelation Function</td>
<td>60</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
<td>3</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
<td>24</td>
</tr>
<tr>
<td>PMF</td>
<td>Probability Mass Function</td>
<td>25</td>
</tr>
<tr>
<td>Q-Q</td>
<td>Quantile-Quantile Plot</td>
<td>25</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
<td>2</td>
</tr>
<tr>
<td>Quake3</td>
<td>Quake III Arena</td>
<td>191</td>
</tr>
<tr>
<td>Quake4</td>
<td>Quake IV</td>
<td>191</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>RTT</td>
<td>Round-trip Time</td>
<td>21</td>
</tr>
<tr>
<td>S2C</td>
<td>Server-to-Client</td>
<td>23</td>
</tr>
<tr>
<td>s2c</td>
<td>server-to-client</td>
<td></td>
</tr>
<tr>
<td>SARIMA</td>
<td>Seasonal Autoregressive Integrated Moving Average</td>
<td>37</td>
</tr>
<tr>
<td>SMAQ</td>
<td>Statistical Match And Queuing tool</td>
<td>37</td>
</tr>
<tr>
<td>SRD</td>
<td>Short-Range Dependent</td>
<td>199</td>
</tr>
<tr>
<td>TES</td>
<td>Transform Expand-Sample</td>
<td>37</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
<td>23</td>
</tr>
<tr>
<td>VG</td>
<td>Variance Gamma</td>
<td>54</td>
</tr>
<tr>
<td>VΓG</td>
<td>Variance Gamma Modified Gaussian</td>
<td>191</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over Internet Protocol</td>
<td>1</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network</td>
<td>9</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
<td>9</td>
</tr>
<tr>
<td>$\lambda^2$</td>
<td>$\lambda^2$ discrepancy measure</td>
<td>25</td>
</tr>
</tbody>
</table>
Appendix B

OMNeT++ code

B.1 Model Parameters

B.1.1 GameBasicApp Parameters

The ‘GameBasicApp.ned’ file, specifies the parameters and the interface required for the traffic model. The values of the parameters for the model are specified in the simulation’s ‘.ini’ file. The parameters required by the ‘GameBasicApp.ned’ module are shown in Table B.1. These parameters are the same as those required by the parent ‘UDPBasicApp’. The parameters specific to the GameBasicApp application are outlined in Table B.2, while the methods are shown in Table B.3.

The methods sendPacket, processPacket and chooseDestAddr have been taken from the ‘UDPBasicApp.cc’, the reader is referred to the INET Framework manual [135] for details of these methods.

The methods of this class should be redefined in the child classes as required by the model for the particular FPS game traffic in question. Usually only the getPayload and getIATime methods need to be changed to incorporate the appropriate functionality for the game being simulated.

To use the traffic generator as a ‘UDP application’ for the INET Framework ‘standard host’, the command: **.udpAppType="GameBasicApp" is included in the simulation’s ‘.ini’ file. The appropriate parameters required by the simulation are also included in this file (refer to [136]).
Table B.1: Parameters for the .ini file

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>localPort</td>
<td>Local port for this application.</td>
</tr>
<tr>
<td>destPort</td>
<td>Destination port for this application.</td>
</tr>
<tr>
<td>destAddresses</td>
<td>List of the destination addresses for the packets.</td>
</tr>
</tbody>
</table>

Table B.2: Parameters for the Game Basic Application

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>Specifies the delay in seconds before the traffic generator starts sending messages</td>
</tr>
<tr>
<td>hostType</td>
<td>Make the traffic generator act as either a ‘client’ or ‘server’ host</td>
</tr>
<tr>
<td>numPlayers</td>
<td>Specifies the number of players participating in the game</td>
</tr>
<tr>
<td>period</td>
<td>Specifies the period of any periodic component in the traffic. 0 implies that no periodic traffic</td>
</tr>
<tr>
<td>pSize</td>
<td>Specifies the payload size of the periodic component</td>
</tr>
</tbody>
</table>

B.1.2 ExGaussArmaApp Parameters

The ARMA traffic simulator is based on the work outlined in Section 7.7, and is implemented in the ‘ExGaussArmaApp.ned’, and ‘ExGaussArmaApp.cc’ files. The ‘ExGaussArmaApp.ned’ file specifies the interface into the traffic generator application. (The reader is referred to Appendix B.5 for details of this file). The main implementation of the model resides in the ‘ExGaussArmaApp.cc’ file (see Appendix B.7). The code for the generator is developed by modifying the appropriate application files available in the INET framework [135].

The ‘ExGaussArmaApp.ned’ is similar to the ‘GameBasicApp.ned’ application, but with an additional parameter that is specific to this application:

- game_type: specifies which game is being modelled. Note that the traffic simulator produces traffic for the seven games in this study and the game traffic model as specified in the IEEE802.16 Evaluation Methodology Document (EMD) [129].

The ‘ExGaussArmaApp.cc’ contains the traffic generator (C++) code for the methods that have been redefined from the ‘GameBasicApp’ class. These methods are shown in Table B.4. For the current implementation, each game is characterised by the parameters shown in Table B.5.
Table B.3: Methods for the Game Basic Application

<table>
<thead>
<tr>
<th>Method</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize</td>
<td>Initialises the variables used by the traffic generator.</td>
</tr>
<tr>
<td>startNow</td>
<td>This method should be redefined in the child classes if there are specific variables that require initialisation.</td>
</tr>
<tr>
<td>handleMessage</td>
<td>Performs the required actions to process a message. If the message is a self-message, this method sends the appropriate number of packets and reschedules the next event. If the host is a server, this method will send the packet to all the clients that are connected to it.</td>
</tr>
<tr>
<td>getPayload</td>
<td>Returns the size of the next message (UDP payload) to send. Current implementation returns a fixed message size of 50 bytes, without the UDP headers. This method should be redefined in child classes to suit the actual game being modelled.</td>
</tr>
<tr>
<td>getIATime</td>
<td>Returns the time interval between messages (ms). Currently this method returns a fixed time of 50 ms. This method should be redefined in child classes to suit the game being modelled.</td>
</tr>
<tr>
<td>extreme</td>
<td>Returns a random number drawn from an Extreme Value distribution specified by location and scale parameters.</td>
</tr>
<tr>
<td>exgauss</td>
<td>Returns a random number drawn from an Ex-Gaussian distribution specified by location, scale and rate parameters.</td>
</tr>
<tr>
<td>gammagauss</td>
<td>Returns a random number drawn from a Gamma Modified Gaussian distribution specified by scale, shape, mean and standard deviation parameters.</td>
</tr>
<tr>
<td>vargammagauss</td>
<td>Returns a random number drawn from a residual generating distribution specified by parameters outlined in Section 7.8.</td>
</tr>
<tr>
<td>finish</td>
<td>Records and prints various user defined statistics upon completion of the simulation.</td>
</tr>
</tbody>
</table>
Table B.4: Methods for the ExGaussArmaApp Application

<table>
<thead>
<tr>
<th>Method</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>startNow</td>
<td>Modified to choose the parameters for the game to be simulated and to initialise the starting values of the various game parameters used by the traffic generator.</td>
</tr>
<tr>
<td>getPayload</td>
<td>The payload is determined using the number of players participating in the game and the parameters of the best fit Ex-Gaussian distribution to the two player game. For the IEEE802.16 EMD model, the payload is drawn from an Extreme Value distribution as specified in [129].</td>
</tr>
<tr>
<td>Client</td>
<td>In Section 5.3.1.8 we found that there was little difference in the fit of various distributions to the payload size distribution, so for simplicity the client payload size is determined from a truncated normal distribution.</td>
</tr>
<tr>
<td>getIATime</td>
<td>Modified so that the appropriate time interval between update packets is determined for the game traffic being simulated. For FPS games that have a single mode in the inter-arrival time distribution, the interval between packets is set by the serverTickTime parameter. For the games with bi-modal inter-arrival time distributions by the serverTickTime and the clientSnapshotRate. For the IEEE802.16 EMD model the interval between packets is drawn from an Extreme Value distribution as specified in [129].</td>
</tr>
<tr>
<td>Client</td>
<td>The interval between packets is determined using the results from Section 4.3.</td>
</tr>
<tr>
<td>ArmaFilter</td>
<td>New method specific to the ExGaussArmaApp class for the ARMA(1,1) filter. This method implements equation 3.87, which is used to determine the size of the next server-to-client update packet based on the previous payload size of the previous packet and the distribution of the residuals. The residuals are drawn from the residual density using the technique described in Section 7.8.</td>
</tr>
</tbody>
</table>

Table B.5: Parameters for the ExGaussArmaApp

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>offset</td>
<td>Estimate of the fixed part of the 2-player Server-to-Client payload.</td>
</tr>
<tr>
<td>muEXG, varEXG, kappaEXG</td>
<td>Parameters of the best fit Ex-Gaussian distribution estimated from a 2-player game as outlined in Section 7.7.</td>
</tr>
<tr>
<td>Mean2Player</td>
<td></td>
</tr>
<tr>
<td>serverTickTime</td>
<td>Parameters defining the Inter-arrival time for the server.</td>
</tr>
<tr>
<td>clientSnapshotRate</td>
<td></td>
</tr>
<tr>
<td>periodicPayloadSize</td>
<td>Parameters specifying the characteristics of periodic traffic. Only one periodic component can be specified in the current implementation.</td>
</tr>
<tr>
<td>periodOfPeriodicComponentServer</td>
<td></td>
</tr>
<tr>
<td>periodOfPeriodicComponentClient</td>
<td></td>
</tr>
<tr>
<td>phi, theta</td>
<td>ARMA(1,1) φ and θ parameters, these are estimated from a 2-player game as outlined in Section 6.2.5.</td>
</tr>
</tbody>
</table>
B.2 GameBasicApp.ned

Removed for copyright reasons

B.3 GameBasicApp.h

Removed for copyright reasons

B.4 GameBasicApp.cc

Removed for copyright reasons

B.5 ExGaussArmaApp .ned

Removed for copyright reasons

B.6 ExGaussArmaApp .h

Removed for copyright reasons

B.7 ExGaussArmaApp .cc

Removed for copyright reasons
Appendix C

Gamma Modified Gaussian Distribution

C.1 Density of the Gamma Modified Gaussian (ΓMG) Distribution

Let $X$ and $Y$ be two independent random variables such that:

$X$ has a Gamma distribution with density $f_1(x)$ and parameters $k, \theta$.

$$f_1(x) = \frac{x^{k-1}e^{-\frac{x}{\theta}}}{\Gamma(k)\theta^k} \quad x \geq 0, \quad k > 0, \quad \theta > 0 \quad (C.1)$$

$Y$ has a Gaussian distribution with density $f_2(y)$ and parameters $\mu, \sigma$.

$$f_2(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} \sigma > 0$$

The distribution of $X + Y$ has a density $g(u, v)$ given by the convolution of the densities of $X$ and $Y$:

$$g(u, v) = \int_{-\infty}^{\infty} f_1(v)f_2(u-v)dv$$

$$g(u, v; \mu, \sigma, k, \theta) = \int_{0}^{\infty} v^{k-1}e^{-\frac{v}{\theta}} \frac{1}{\Gamma(k)\theta^k}\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(u-v-\mu)^2}{2\sigma^2}}dv$$

$$= \frac{1}{\Gamma(k)\theta^k\sqrt{2\pi\sigma^2}} \int_{0}^{\infty} v^{k-1}e^{-\frac{v}{\theta}} e^{-\frac{(u-v-\mu)^2}{2\sigma^2}}dv$$

let $\alpha = u - \mu$ and $p = \frac{1}{2\sigma^2}$ with $p > 0$ then:
C Gamma Modified Gaussian Distribution

\[ g(\alpha, v; k, \theta, p) = \frac{\sqrt{p}}{\Gamma(k) \theta^k \sqrt{\pi}} \int_0^\infty v^{k-1} e^{-\frac{1}{2} \theta v^2 - \frac{1}{2} p(\alpha - v)^2} dv \]

\[ = \frac{\sqrt{p}}{\Gamma(k) \theta^k \sqrt{\pi}} \int_0^\infty v^{k-1} e^{-p(\alpha - v)^2 - \frac{1}{2} \theta v^2} dv \]

\[ = \frac{\sqrt{p}}{\Gamma(k) \theta^k \sqrt{\pi}} \int_0^\infty v^{k-1} e^{-p(\alpha - v)^2 - \frac{1}{2} \theta v^2} dv \]

\[ = \frac{\sqrt{p} e^{-p\alpha^2}}{\Gamma(k) \theta^k \sqrt{\pi}} \int_0^\infty v^{k-1} e^{-p\alpha^2 - \frac{1}{2} (1 - 2p\alpha v - p\theta v^2)} dv \]

\[ = \frac{\sqrt{p} e^{-p\alpha^2}}{\Gamma(k) \theta^k \sqrt{\pi}} \int_0^\infty v^{k-1} e^{-p\alpha^2 - \frac{1}{2} (1 - 2p\alpha) v^2} dv \]

let \( q = \frac{1}{\theta} - 2p\alpha \) or \( \alpha = \frac{1 - q\theta}{2\mu} \) thus:

\[ g(q, v; k, \theta, p) = \frac{\sqrt{p} e^{-\frac{1 - 2q\theta + \frac{q^2\theta^2}{2}}{4\mu^2 \theta^2}}}{\Gamma(k) \theta^k \sqrt{\pi}} \int_0^\infty v^{k-1} e^{-pv^2 - qv} dv \] \hspace{1cm} (C.2)

This integral has a solution of the form [109, p. 343]:

\[ \int_0^\infty v^{k-1} e^{-pv^2 - qv} dv = \Gamma(k) (2p)^{\frac{k}{2}} e^{\frac{q^2}{4p}} D_{-k} \left( \frac{q}{\sqrt{2p}} \right) \text{ for } k \text{ Real and } \text{Real}(p) > 0 \]

where \( D_{-k} \left( \frac{q}{\sqrt{2p}} \right) \) is a parabolic cylinder function.

Substituting for the integral in equation C.2 results in:

\[ g(q; k, \theta, p) = \frac{\sqrt{p} e^{-\frac{1 - 2q\theta + \frac{q^2\theta^2}{2}}{4\mu^2 \theta^2}}}{\Gamma(k) \theta^k \sqrt{\pi}} \Gamma(k) (2p)^{\frac{k}{2}} e^{\frac{q^2}{4p}} D_{-k} \left( \frac{q}{\sqrt{2p}} \right) \]

\[ = \frac{e^{-\frac{1 - 2q\theta + \frac{q^2\theta^2}{2}}{4\mu^2 \theta^2}}}{\theta^k \sqrt{\pi}} \frac{2^{-k} p^{\frac{1}{2} k} \Gamma(1 - k)}{\Gamma(k - k)} e^{\frac{q^2}{4p}} D_{-k} \left( \frac{q}{\sqrt{2p}} \right) \]

\[ = \frac{e^{-\frac{1 - 2q\theta + \frac{q^2\theta^2}{2}}{4\mu^2 \theta^2}}}{\theta^k \sqrt{\pi}} \frac{2^{-k} p^{\frac{1}{2} k} \Gamma(1 - k)}{\Gamma(k - k)} e^{\frac{q^2}{4p}} D_{-k} \left( \frac{q}{\sqrt{2p}} \right) \]

\[ g(q; k, \theta, p) = \frac{1}{\theta^k \sqrt{\pi}} 2^{-\frac{k}{2} p} \frac{1}{\Gamma(k - k)} e^{-\frac{1 - 2q\theta + \frac{q^2\theta^2}{2}}{4\mu^2 \theta^2}} D_{-k} \left( \frac{q}{\sqrt{2p}} \right) \]

\[ g(q; k, \theta, p) = 2^{-\frac{k}{2} p} \frac{1}{\Gamma(k - k)} e^{-\frac{1 - 2q\theta + \frac{q^2\theta^2}{2}}{4\mu^2 \theta^2}} D_{-k} \left( \frac{q}{\sqrt{2p}} \right) \] \hspace{1cm} (C.3)

Equation C.3 is a three parameter \((k, \theta, p)\) Probability Density Function of the GEG distribution in terms of the random variable \( q \). Expressing this Probability Density Function in terms of the original parameters \( \mu, \sigma \) and variable \( u \) leads to:
C.1 Density of the ΓMG Distribution

\[ g(q; k, \theta, \sigma) = \frac{2^{-k} \Gamma \left( \frac{1}{2k} \right)}{\theta^{k} \sqrt{\pi}} e^{-\frac{\left( \frac{q}{\sigma} \right)^{2}}{k^{2}}} D_{-k} \left( \frac{q}{\sqrt{2} \sigma} \right) \]

\[ g(q; k, \theta, \sigma) = \frac{\sigma^{k-1}}{\theta^{k} \sqrt{2\pi}} e^{-\frac{\left( \frac{q}{\sigma} \right)^{2}}{k^{2}}} D_{-k} (q\sigma) \]

\[ g(q; k, \theta, \sigma) = \frac{\sigma^{k-1}}{\theta^{k} \sqrt{2\pi}} e^{-\frac{\left( \frac{q}{\sigma} \right)^{2}}{k^{2}}} D_{-k} (q\sigma) \]

\[ g(\alpha; k, \theta, \sigma) = \frac{\sigma^{k-1}}{\theta^{k} \sqrt{2\pi}} e^{-\frac{\left( \frac{\alpha}{\sigma} \right)^{2}}{k^{2}}} D_{-k} \left( \frac{\sigma^{2} - \alpha \theta}{\sigma \theta} \right) \]

\[ g(\alpha; k, \theta, \sigma) = \frac{\sigma^{k-1}}{\theta^{k} \sqrt{2\pi}} e^{-\frac{\left( \frac{\alpha}{\sigma} \right)^{2}}{k^{2}}} D_{-k} \left( \frac{\sigma^{2} - \alpha \theta}{\sigma \theta} \right) \]

\[ g(u; k, \theta, \sigma) = \frac{\sigma^{k-1}}{\theta^{k} \sqrt{2\pi}} e^{-\frac{\left( \frac{u}{\sigma} - \mu \right)^{2}}{k^{2}}} D_{-k} \left( \frac{\sigma^{2} - (u - \mu) \theta}{\sigma \theta} \right) \quad (C.4) \]

Equation C.4 is a four parameter representation of the PDF of the ΓMG distribution.

An alternative expression can be obtained by expressing the parabolic cylinder function in terms of repeated integrals of the complementary error function see Abramowitz and Stegun [1, p. 300].

\[ D_{-n-1} \left( \frac{z}{\sqrt{2}} \right) = e^{\frac{z^{2}}{2}} \sqrt{2^{n-1} \pi} I^{n} \text{erfc}(z) \]

where \( I^{n} \text{erfc}(z) \) represents \( n \) repeated integrals of the \( \text{erfc}(z) \)

let \( z/\sqrt{2p} = \frac{q}{2\sqrt{p}} \) \( z^{2} = \frac{q^{2}}{4p} \)

\[ D_{-k} \left( \frac{q}{\sqrt{2p}} \right) = e^{\frac{q^{2}}{2p}} \sqrt{2^{k-2} \pi} I^{k-1} \text{erfc}(\frac{q}{2\sqrt{p}}) \]

Substituting in equation C.3 for the parabolic cylinder function results in:

\[ g(q; k, \theta, p) = \frac{2^{-k} \Gamma \left( \frac{1}{2k} \right)}{\theta^{k} \sqrt{\pi}} e^{-\frac{\left( \frac{2 \sqrt{2} \theta \sqrt{p} + q^{2} \theta^{2}}{4\theta^{2}} \right)}{k^{2}}} \sqrt{2^{k-2} \pi} I^{k-1} \text{erfc}(\frac{q}{2\sqrt{p}}) \]

\[ g(q; k, \theta, p) = \frac{p^{\frac{1-k}{2}}}{2\theta^{k}} e^{-\frac{\left( \frac{q^{2}}{4\theta^{2}} \right)}{k^{2}}} \sqrt{2^{k-2} \pi} I^{k-1} \text{erfc}(\frac{q}{2\sqrt{p}}) \]
\[ g(q; k, \theta, \sigma) = \frac{2^{k-\frac{3}{2}} \sigma^{1-k}}{\theta^k} e\left(-\frac{\sigma^2 - \theta^2}{2\sigma^2}\right) I_{k-1} e^{f c}\left(\frac{q\sigma}{\sqrt{2}}\right) \]

\[ g(\alpha; k, \theta, \sigma) = \frac{2^{k-\frac{3}{2}} \sigma^{1-k}}{\theta^k} e\left(-\frac{\sigma^2 + \frac{3(\frac{1}{2} - \frac{2k}{\alpha})}{\sigma^2}}{2\sigma^2}\right) I_{k-1} e^{f c}\left(\frac{\frac{1}{2} - \frac{\alpha}{\sigma^2}}{\sqrt{2}}\right) \]

\[ g(\alpha; k, \theta, \sigma) = \frac{2^{k-\frac{3}{2}} \sigma^{1-k}}{\theta^k} e\left(-\frac{\sigma^2 + \frac{3(\frac{1}{2} - \frac{2k}{\alpha})}{\sigma^2}}{2\sigma^2}\right) I_{k-1} e^{f c}\left(\frac{\alpha^2 - \theta^2}{2\sigma^2}\right) \]

or

\[ g(\alpha; k, \theta, \sigma) = \frac{2^{k-\frac{3}{2}} \sigma^{1-k}}{\theta^k} e\left(\frac{\sigma^2 - \frac{\alpha^2 - \theta^2}{\sigma^2}}{2\sigma^2}\right) I_{k-1} e^{f c}\left(\frac{\alpha^2 - \theta^2}{2\sigma^2}\right) \]

\[ g(\alpha; k, \theta, \sigma) = \frac{2^{k-\frac{3}{2}} \sigma^{1-k}}{\theta^k} e\left(\frac{\sigma^2 - \frac{\alpha^2 - \theta^2}{\sigma^2}}{2\sigma^2}\right) I_{k-1} e^{f c}\left(\frac{\alpha^2 - \theta^2}{2\sigma^2}\right) \]

\[ g(\alpha; k, \theta, \sigma) = \frac{2^{k-\frac{3}{2}} \sigma^{1-k}}{\theta^k} e\left(\frac{\sigma^2 - \frac{\alpha^2 - \theta^2}{\sigma^2}}{2\sigma^2}\right) I_{k-1} e^{f c}\left(\frac{\alpha^2 - \theta^2}{2\sigma^2}\right) \]

Equation C.5 is a four parameter representation of the PDF of the \( \Gamma \)MG distribution.

C.2 Density of the Ex-Gaussian Distribution

In the case where the shape parameter of the gamma distribution \( k \) in equation C.1 is unity, the gamma distribution becomes the exponential distribution. In this case equation C.5 can be written as:

\[ g(u; 1, \theta, \sigma, \mu) = \frac{1}{2\theta} e\left(\frac{\sigma^2 - (u - \mu)\theta}{\sigma\theta}\right) I_{0} e^{f c}\left(\frac{\sigma^2 - (u - \mu)\theta}{\sqrt{2}\sigma\theta}\right) \]

since \( e^{f c}(z) = 2\Theta\left(-z\sqrt{2}\right) \) where \( \Theta(z) \) is the standard Gaussian cumulative distribution function given by Abramowitz and Stegun [1, p. 298]:

\[ \Theta(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{2} e^{-\frac{x^2}{2}} dx \]

in this case \( z = \frac{\sigma^2 - (u - \mu)\theta}{\sqrt{2}\sigma\theta} \) so \( -z\sqrt{2} = -\frac{\sigma^2 - (u - \mu)\theta}{\sigma\theta} = \frac{(u - \mu) - \frac{\sigma^2}{\sigma}}{\sigma} \)

\[ g(u; 1, \theta, \sigma, \mu) = \frac{1}{2\theta} e\left(\frac{\sigma^2 - (u - \mu)\theta}{\sigma\theta}\right) 2\Theta\left(\frac{(u - \mu) - \frac{\sigma^2}{\sigma}}{\sigma}\right) \]

\[ g(u; 1, \theta, \sigma, \mu) = \frac{1}{\theta} e\left(\frac{\sigma^2 - (u - \mu)\theta}{\sigma\theta}\right) \Theta\left(\frac{(u - \mu) - \frac{\sigma^2}{\sigma}}{\sigma}\right) \]

let \( \lambda = \frac{1}{\theta} \)
C.2 Density of the Ex-Gaussian Distribution

\[ g(u) = \lambda \exp \left( \frac{u - \mu}{\sigma} + \frac{\lambda^2}{2} \right) \Theta \left( \frac{(u - \mu) - \lambda \sigma^2}{\sigma} \right) \]

\[ g(u; \lambda, \sigma, \mu) = \lambda \exp \left( \lambda (\mu - u) + \frac{\lambda^2}{2} \right) \Theta \left( \frac{u - \mu}{} - \lambda \sigma^2 \right) \]  \hspace{1cm} (C.6)

Equation C.6 is a three parameter representation of the PDF of the EG distribution.