Creation and Distribution of Real-time Content: A Case Study in Provisioning Immersive Voice Communications to Networked Games.

A thesis submitted in fulfilment of the
requirements for the award of the degree

Doctor of Philosophy

from

THE UNIVERSITY OF WOLLONGONG

by

Cong Duc Nguyen
Bachelor of Engineering (Honours Class I)

SCHOOL OF ELECTRICAL, COMPUTER
AND TELECOMMUNICATIONS ENGINEERING
2006

Abstract

The rapid increases in network bandwidth and processing power have led to tremendous growth in Internet applications and changed the nature of delivering content. From earlier web application, which is only about retrieval of pre-computed static content, current content delivery architectures provide dynamic content and enable personalization of content. Recent interactive entertainment applications, such as multiplayer online games, require content to be created and distributed in real-time. In addition, the rapid increase in processor speed has led to various proposals to put real-time computation within or at the edge of the network that allow application specific processing on packet flow such as multimedia transcoding and content adaptation. In short, these emerging applications have a common characteristics that the contents of application flows are processed in real-time before being delivered to end users. We refer to these as applications that require *real-time content creation*.

This thesis aims to develop models for real-time content creation and distribution. We examine both network architectures for delivering content as well as server processing resource management for content creation. We concentrate on a case study when content creation is from a dynamic set of dispersed sources. In particular, we examine the provision of an immersive voice communication service to massively multiplayer online games, which requires real-time creation of audio scenes from dynamic sets of participants. We present various delivery architectures for this service, evaluate the performance of these architectures and provide recommendations based on the evaluation. In addition, this thesis designs a server resource management architecture for sharing processing resource among real-time content creation applications.

Our study begins with reviewing the evolution of content distribution over the Internet, ranging from simple caching proxies to content distribution networks and personalization of content. We then discuss current and future developments of content distribution which require real-time creation and distribution of content from dynamic sets of dispersed sources. These include *state information* processing and *communication information* processing in distributed virtual environments. While all networked games require state information processing, communication information processing has been recently seen as a key to enhance the reality and attractiveness of the virtual environment. In particular, this thesis reviews technologies and approaches for providing an immersive voice communication service to distributed virtual environments. Several delivery architectures are introduced for providing this service, namely peer-to-peer, central server, distributed locale server architecture, and distributed proxy architecture. Furthermore, we present a realistic simulation model that captures player distribution in the Internet and avatar distribution in the game virtual world and specify two key performance evaluation parameters: interactive delay and network bandwidth usage.

In the central server architecture, two optimization objectives are proposed for choosing an optimal central server from a set of potential servers. We also propose a dynamic relocation of a central server in response to changes in player distribution due to time zone differences. It is shown that relocation of the central server in response to these changes can significantly reduce the interactive delay by up to 40% and the network bandwidth usage by up to 50%. In addition, the optimal central server can significantly reduce the interactive delay compared to a randomly located central server.

In the distributed locale server architecture, the game virtual world is partitioned into smaller areas called locales and each locale is assigned to a server. We propose two server assignment algorithms for optimizing the latency performance of this architecture. The first algorithm is based on an Integer Linear Programming (ILP) model which provides an exact solution to the problem but is subject to high computation complexity. We then produce a new multi-layer graph representation of the problem and devise a greedy heuristic based on this graph. It is shown that the greedy heuris-

tics has low run time complexity and provides solutions close to the optimal (within 5% of the optimal in all cases). In addition, increasing the number of servers reduces the latency of the distributed locale server architecture significantly compared to the optimal central server when there is a physical/virtual world correlation. Specifically, with a reasonable number of servers, the distributed locale server architecture can reduce the delay of the central server by 20% to 60%.

In the distributed proxy architecture, players are assigned to a close proxy and each proxy manages the audio mixing operation on behalf of players and forwards audio streams from players to other interested proxies. This thesis develops an ILP model for an optimal proxy assignment and adapts the multi-layer graph approach used earlier to devise a greedy heuristics for solving the proxy assignment problem efficiently. While the ILP model is unscalable, the greedy heuristics is highly scalable and suitable for practical implementation. This thesis also investigates the efficiency of network multicast in different player and avatar distribution scenarios. The effect of varying the number of proxies is also investigated.

Extensive simulation experiments are carried out to evaluate the performance of all delivery architectures. In particular, since the distributed locale server architecture and the distributed proxy architecture are 'dual' of each other, we concentrate on comparing the performance of these. From the performance evaluation, we provide recommendations on choosing suitable delivery architectures based on the server resource availability, multicast, and game's avatar aggregation behaviors. The quantitative study in this thesis will be of benefit to future immersive voice service providers in the design of a cost effective delivery architecture for this service.

Finally, the thesis presents a resource management architecture for sharing processing resources among various real-time content creation applications including the immersive audio mixing application. Due to the inability of determining processing times for scheduling, a processing resources scheduling algorithm called Start-time Weighted Fair Queueing (SWFQ) is proposed. From analysis and simulation, it is shown that SWFQ offers good fairness and delay properties compared to current schemes. In fact, the fairness of SWFQ was comparable to Weighted Fair Queueing (WFQ) and the delay behavior is better than Start-time Fair Queueing (SFQ).

Statement of Originality

This is to certify that the work described in this thesis is entirely my own work, except where due reference is made in the text. I also acknowlege the guidance from my supervisors and ideas generated from discussions with them in this work.

No work in this thesis has been submitted for a degree to any other university or institution.

Signed

Cong Duc Nguyen
1 May, 2006

Acknowledgments

First of all, I would like to thank my supervisor Professor Farzad Safaei for his guidance, support and encouragements during my PhD, especially through some difficult stages of the project. I am also very grateful to Dr. Paul Boustead for his help and support. This thesis may have not been completed without their help.

Next, I would like to express my gratitude to Professor Joe Chicharo for giving me the opportunity to undertake this study. I am also grateful to Smart Internet Technology Cooperative Research Center for supporting this work. I would like to thank Dr. Don Platt for his assistance in the early stages of the project and Fariza Sabrina for her collaboration.

I would like to thank all members of TITR lab for their encouragements and assistance. In particular, I would like to thank Vinh Nguyen, Ying Que, Jeremy Brun, Daniel Franklin and Justin Lipman.

I would like to thank Lan Nguyen, Cuong Tran and Long Nghiem for their friendship and support during my PhD in Wollongong.

Finally, I would like to thank my family and Hang Thuy Nguyen for their encouragements and support throughout my PhD.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Overview 3

1.3 Contributions 7

1.4 Publications based on Thesis 8

2 Literature Review **10**

2.1 Introduction 10

2.2 Evolution of Content Delivery 11

2.2.1 Proxy Caching 12

2.2.2 Web Server Cluster 13

2.2.3 Content Distribution Networks 14

2.2.4 Advanced Content Service Delivery 16

2.3 Future Development of Content Creation and Delivery 19

2.3.1 Multiplayer Online Games 19

2.3.2 Group Communications in Multiplayer Online Games . . . 24

2.3.3 Immersive Voice Communication in Distributed Virtual Environments 27

2.3.4 Audio Codec Technologies 32

2.4 Infrastructure Support 37

2.4.1 Server Infrastructure 37
2.4.2 Network Infrastructure Support 45
2.5 Conclusions 55
2.5.1 Issues Considered in Thesis 55

3 An Immersive Audio Communication Service for Multi-Player Online Games 57

3.1 Introduction 57
3.2 Concept of Immersive Voice Communication Service 58
3.3 Basic Delivery Architectures 61
3.3.1 Peer-to-peer 61
3.3.2 Central Server 62
3.3.3 Distributed Locale Servers 64
3.3.4 Distributed Proxies 66
3.3.5 Discussion on peer-to-peer and server architectures 67
3.4 Framework for Performance Evaluation 68
3.4.1 Game Player Grouping Behaviors 68
3.4.2 Models of Physical Networks and Virtual World 71
3.4.3 Definition of Parameters and Assumption 75
3.5 Conclusions 77

4 Central Server 78

4.1 Introduction 78
4.2 Service Delivery Model 79
4.2.1 Optimization Procedures 79
4.2.2 Relocation of a Central Server 81
4.3 Simulation Experiments 84

4.3.1 Simulation Setup 84
4.3.2 Relocation of a Central Server 85
4.3.3 Comparison of Optimization Objectives 87
4.3.4 Effect of Varying Physical/Virtual World Correlation on Network Resources and Delay Metrics 88
4.4 Conclusions 90

5 Distributed Locale Server 91

5.1 Introduction 91
5.2 Service Delivery Model 92
5.2.1 Problem Description 93
5.2.2 Mathematical Programming Model 94
5.2.3 Greedy Heuristic Algorithm 96
5.2.4 Impact of Avatar Movements and Player Distribution on Optimal Server Assignment 100
5.3 Simulation Experiments 100
5.3.1 Simulation Setup 100
5.3.2 Investigation of Server Assignment Algorithms 101
5.3.3 Effect of Varying Number of Servers and Physical/Virtual World Correlation 103
5.3.4 Effect of Varying Correlation in Interactive Delay 105
5.3.5 Network Bandwidth Requirements in Different Avatar Aggregation Behaviors 108
5.4 Conclusions 110

6 Distributed Proxy Architecture 112

6.1 Introduction 112
6.2 Service Delivery Model 114

6.2.1 Proxy Location Problem 114
6.2.2 Mathematical Programming Model 115
6.2.3 Heuristic Algorithms 118
6.2.4 Different Proxy Architectures 121
6.3 Simulation Experiments 123
6.3.1 Simulation Setup 123
6.3.2 Investigation of Proxy Assignment Algorithms 124
6.3.3 Investigation with Proxy Architectures for Different Player Aggregation Behaviours 128
6.3.4 Efficiency of Multicast 132
6.3.5 Effect of Varying Number of POPs and Servers 134
6.4 Conclusions 136

7 Comparison of Architectures 138

7.1 Introduction 138
7.2 Comparisons of Architectural Requirements 139
7.2.1 Server Assignment Algorithms 139
7.2.2 Impact of Avatar Movements and Player Distribution 141
7.2.3 Server Resource Requirements 142
7.3 Simulation Experiments 142
7.3.1 Simulation Setup 142
7.3.2 Interactive Delay 143
7.3.3 Network Bandwidth Usage 148
7.4 Summary of Results and Recommendations 151
7.4.1 Impact of Avatar Aggregations on Delivery Architectures . . 151
7.4.2 Impact of Number of Servers 152

7.4.3 Impact of Correlation 152
7.4.4 Efficiency of Multicast 153
7.4.5 Discussions on Choice of Delivery Architectures 153

8 Server Processing Resource Management 156

8.1 Introduction 156
8.2 Server Processing Resource Management 158
8.2.1 Processing Resource Requirements of Immersive Audio Mixing Operation 158
8.2.2 Server Processing Resource Management Model 159
8.2.3 Processor Resource Scheduling 161
8.3 Start-time Weighted Fair Queueing 162
8.3.1 Packet Scheduling Disciplines in Traditional Networks . . . 164
8.3.2 Server Processing Resource Scheduling Model 165
8.3.3 Example of SWFQ 168
8.4 Analysis of SWFQ 169
8.4.1 Fairness Analysis 169
8.4.2 Delay Analysis 170
8.5 Simulation Experiments 172
8.5.1 Simulation Setup 172
8.5.2 Fairness of SWFQ 173
8.5.3 Delay Properties of SWFQ 173
8.6 Conclusions 178

9 Conclusions 183

9.1 Overview 183
9.2 Summary of Contributions and Findings 183

9.2.1 Classification of Architectures and Performance Evaluation Framework 184
9.2.2 Central Server 184
9.2.3 Distributed Locale Servers 185
9.2.4 Distributed Proxy Architecture 185
9.2.5 Performance Comparison Evaluation 186
9.2.6 Server Resource Management 186
9.3 Thesis Recommendations 187
9.3.1 Recommendations on Infrastructure Support 187
9.3.2 Recommendations on Delivery Architectures 188
9.4 Future Work 189
9.4.1 Performance Evaluation Model 189
9.4.2 Experimental Investigation 191

A Proofs for SWFQ Analysis 204

A.1 Proof of Theorem 1 204
A.2 Proof of Theorem 2 205
A.3 Proof of Theorem 3 206
A.4 Proof of Theorem 4 207

B Details of Simulation Environments 209

B.1 Network Topologies 209
B.2 Physical Network and Virtual World 212
B.2.1 Physical Networks 212
B.2.2 Virtual World 213
B.3 Simulation Procedures 214

List of Figures

2.1 A content distribution network. 14

2.2 Content delivery/assembly using Edge Side Includes (ESI). 17

2.3 Example of an ESI template consisting of ESI fragments and their expiration times. 18

2.4 Example of ICAP operation. 18

2.5 Game server architectures. 21

2.6 Example of a game virtual world 24

2.7 Basic functional elements of DICE, adapted from (Boustead et al., 2005). 31

2.8 Angular clustering, adapted from (Boustead et al., 2005). 32

2.9 The layered Grid architecture (Foster and Kesselman, 2004). 40

2.10 Example of IP multicast and application layer multicast. 48

2.11 Illustration of a service overlay network, adapted from (Duan et al., 2003). 49

2.12 Booster box and the deployment of booster boxes for distributed game servers, adapted from (Rooney et al., 2003). 52

2.13 Overlay server and routing between these servers over multiple network domains (Boustead et al., 2004). 53

2.14 Tunnel Switch design (Boustead et al., 2004). 54

3.1 Immersive voice communication scenario and zone definition 59

3.2 Peer-to-peer architecture for immersive audio scene creation 63

3.3 Central server architectures for immersive audio scene creation . . . 64

3.4 Distributed locale server architecture for immersive audio scene creation . . . 65

3.5 Distributed proxy server architecture for immersive audio scene creation . . . 67

3.6 Player behavior classification . . . 71

3.7 Avatar distribution in different games . . . 73

3.8 Delay components. . . . 76

3.9 Interactive delay and bandwidth cost metrics associated with avatar "a". . . . 76

4.1 Physical/virtual world model. . . . 81

4.2 Relocation of a central server during a transient period. . . . 83

4.3 Effect of changes in game client distribution on the interactive delay metric and network bandwidth usage of a fixed central server versus the optimal central server. . . . 86

4.4 Interactive delay comparison of the two optimization objectives in different cluster distribution. . . . 86

4.5 Effect of varying physical/virtual world correlation on network resource usage of multicast and peer-to-peer unicast versus the central server architecture. . . . 88

4.6 Effect of varying physical/virtual world correlation on the interactive delay metric . . . 89

5.1 Graph representation of the server assignment problem and solution. 98

5.2 Server assignment results from Cplex and the greedy heuristics. . . . 103

5.3 Effect of changes in number of server and physical/virtual world correlation. . . . 104

5.4 Effect of changes in correlation on interactive delay for crowd based games. . . . 106

5.5 Effect of changes in correlation on interactive delay for clan based games. . . . 107

5.6 Bandwidth resource requirements of distributed server and peer-to-peer versus central server for crowd/clan based games 109

5.7 Network bandwidth requirements in a loner based game 110

6.1 Graph representation of the proxy assignment problem and solution. 119

6.2 Network bandwidth usages and interactive delay calculation associated with avatar “a”. 122

6.3 Comparison of proxy assignment algorithms. 127

6.4 Effect of varying virtual/physical world correlation on network bandwidth requirements in crowd/clan based games 130

6.5 Effect of varying density on network bandwidth requirements in loner and crowd based games 131

6.6 Effect of varying the number of POPs on network bandwidth requirements of proxy multicast and unicast 133

6.7 Effect of varying virtual/physical world correlation on interactive delays 134

6.8 Effect of varying number of proxies on interactive delay and network bandwidth requirements of distributed proxy architectures 135

7.1 Comparison in interactive delays between distributed proxies and distributed locale servers when varying the physical/virtual world correlation. 144

7.2 Comparison in interactive delays between distributed proxies and distributed locale servers when varying the physical/virtual world correlation. 145

7.3 Comparison in interactive delays between distributed proxies and distributed locale servers when varying the number of server. 146

7.4 Network bandwidth usages of distributed proxies in clan/crowd based games when varying the physical/virtual world correlation. 149

7.5 Network bandwidth usages of distributed locale servers in clan/crowd based games when varying the physical/virtual world correlation. . . 150

7.6 Effect of avatar density on network bandwidth usages of different architectures 155

8.1 An example of implementation of SWFQ in SWON. 158

8.2 Immersive Audio Mixing. 160

8.3 Variation in actual processing time requirements of immersive audio scene of a Quake 3 game with five players. 160

8.4 Model of a node processing resource management. 161

8.5 Variation in actual processing time of MPEG2 data block of fixed length for different executions (Sabrina and Jha, 2003). 163

8.6 Example of WFQ, SWFQ and SFQ. 167

8.7 Processing rates allocated to IP Forwading, Cast Encryption, and FEC. 174

8.8 Maximum delays of packets in IP Forwarding (queues: 0-9), Cast Encryption (queues: 10-19), and FEC (queues:20-29). 176

8.9 Delay performance of SWFQ and SFQ with Forward Error Coding application. 177

8.10 Delay performance of SWFQ and SFQ with Audio mixing applications. 179

8.11 Delay performance of SWFQ and SFQ with RC2 Encryption application. 180

8.12 Delay performance of SWFQ and SFQ with MPEG2 Encoding application. 181

B.1 Example of (a) random graph and (b) transit-stub graph. 210

List of Tables

2.1 Summary of common audio codecs and the codec-related processing delays in IP-applications (one frame per packet), specified in ITU-T G.114. 35

2.2 Bandwidth requirements for several common VoIP audio codecs. . . 36

4.1 Game client distribution in a day period 85

5.1 Comparison between optimal results and greedy heuristic results in the total interactive communication delay. 102

6.1 Comparison between the optimal proxy assignment, the greedy heuristics, and the simple heuristics with respect to total interactive communication delay. 125

6.2 Comparison between optimal results and greedy heuristic results in the total interactive communication delay. 128

6.3 Comparison between the greedy heuristics and the simple heuristics in the total interactive communication delay. 128

6.4 Comparison between the greedy heuristic and the simple heuristic in the total interactive communication delay. 129

6.5 Comparison between the greedy heuristics and the simple heuristics in the total interactive communication delay. 129

8.1 Delay standard deviations of SWFQ and SFQ with IP Forwarding, Cast Encryption and Solomon Forward Error Coding (msec). 175

8.2 Processing requirements of Audio Mixing, RC2 Encryption and MPEG2 Encoding. 175

8.3 Delay standard deviations of SWFQ and SFQ with Audio Mixing, RC2 Encryption and MPEG2 Encoding (msec). 178

B.1 Parameters used for generating the transit-stub graph. 212

List of Abbreviations

ADPCM	Adaptive Differential Pulse Code Modulation
AS	Autonomous System
API	Application Program Interface
ASIC	Application-Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
CDN	Content Distribution Network
CPU	Central Processing Unit
CVE	Collaborative Virtual Environment
DiffServ	Differentiated Service
DNS	Domain Name Server
FEC	Forward Error Correction
FPS	First Person Shooter
FPGA	Field Programmable Gate Array
FIB	Forwarding Information Base
GPS	Generalized Processor Sharing
GT-ITM	Georgia Tech Internet Topology Model
HRTF	Head Related Transfer Function
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IP	Internet Protocol
IPSec	IP Security Protocol
ISP POP	Internet Service Provider Point of Presence

LAN	Local Area Network
LPC	Linear Predictive Coding
LDD	Latency Driven Distribution
MMOG	Massively Multi-Player Online Games
MPLS	Multi-Protocol Label Switching
NP	Network Processor
NS-2	Network Simulator version 2
OSPF	Open Shortest Path First
PGPS	Packet Generalized Processor Sharing
PVN	Programmable Virtual Network
QoS	Quality of Service
RDD	Resource Driven Distribution
RFC	Request For Comment
RON	Resilient Overlay Network
RSVP	Resource Reservation Protocol
RTP	Real Time Transport Protocol
SFQ	Start-Time Fair Queueing
SON	Service Overlay Network
SPF	Shortest Path First
SIP	Session Initiation Protocol
SWFQ	Start-Time Weighted Fair Queueing
SWON	Switched Overlay Network
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VO	Virtual Organization
VPN	Virtual Private Network
WAN	Wide Area Network
WFQ	Weighted Fair Queueing

Chapter 1

Introduction

1.1 Background

In the last two decades, the Internet has grown tremendously. In addition to the huge increase in the network bandwidth and the number of users, the most noticeable development of the Internet is the evolution of applications. From simple applications such as FTP and email, the Internet now supports various applications ranging from the world wide web and voice over IP to video streaming and multiplayer games. Especially, there is a convergence between the Internet and entertainment applications. Entertainment applications such as multiplayer online games have become very popular and are growing rapidly.

In addition, the nature of content delivery over the Internet has changed significantly. From earlier web applications, which is only about retrieval of pre-computed static content, current content delivery architectures are capable of providing dynamic content and enabling personalization of content. Recent interactive applications, such as multiplayer online games, require creation and distribution of content in real-time. In addition, the rapid increase in processor speeds has also led to various proposals to put more real-time computation within the network for processing the content of application flows. This allows application specific processing on the packet flow such as multimedia transcoding and content adaptation.

The new applications have a common characteristic that the contents of application

flows are processed in real-time before being delivered to end users. We refer to these as applications that require *real-time creation of content*. In some cases, an application only requires a content processing server, such as a multimedia transcoding server, between a pair of source and destination. In other cases, such as multiplayer online games, the processed content, such as *state information*, is from a dynamic set of dispersed sources. The latter case is the focus of this thesis.

In this thesis, we concentrate on a particularly challenging class of applications that require real-time composition of interactive multimedia content from a dynamic set of dispersed sources. Our aim is to design a suitable server and network infrastructure to enable these applications and scale them to a large number of users. To successfully deploy these applications, there are several issues to be considered including *latency*, *network bandwidth*, and *processing* requirements. Among these issues, 'managing latency' is the key one due to the requirement of interactive communication.

In this thesis, we present a case study on the provision of an immersive voice communication service for multiplayer online games that require real-time processing of audio streams from players. We envisage that there are potential server sites located in various geographical regions for the provision of this service. There is also network infrastructure support for connecting these servers. For example, these servers can be connected by low latency and uncongested routing paths in order to improve the application delay performance. This thesis provides mechanisms to deploy these servers in a way that improves application delay performance as well as network bandwidth resources. In addition, each server may be shared by several applications. There is a need to provide processing resource sharing mechanisms between these applications.

This thesis aims to investigate the provision of emerging applications that require real-time creation and distribution of content over the Internet. In particular, the main objectives of the thesis are:

- Examine network and server architectures for the provision of an immersive voice communication service to multiplayer online games and provide suitable

server assignment algorithms for each architecture with a focus on *minimizing latency*.

- Evaluate the performance of these delivery architectures with a focus on voice communication in massively multiplayer online games.
- Design a resource management architecture for sharing processing resources among applications that require real-time content creation at a server.

1.2 Overview

This dissertation investigates the real-time creation and distribution of content over the Internet. The investigation focuses on a particular application, which is real-time processing of audio streams for providing immersive voice communications to multi-player online games. However, the results from the thesis have wider applicability. Chapter 2 reviews the related work on content creation and distribution over the Internet and network and server infrastructures to support those. In addition, we review multiplayer online games, voice communication methods in current games, and research efforts in providing immersive voice communication in distributed virtual environments. In Chapter 3, we introduce the concept of an immersive voice communication service for multiplayer online games and present different delivery architectures for this services. We also present a simulation model framework that captures the game service delivery. This is then followed by three chapters that examine different delivery architectures. Chapter 4 investigates the use of a central server for this service. In Chapter 5, we propose two server assignment algorithms for a distributed server architecture and evaluate the performance of these algorithms. Chapter 6 examines the use of distributed proxies for this service and presents two proxy assignment algorithms. An overall performance evaluation of these architectures is presented in Chapter 7. Chapter 8 presents an architecture for sharing processing resources between the immersive audio mixing and other applications at a server. The chapter also proposes a novel scheduling algorithm for this purpose. Chapter 9 concludes the thesis. The remainder of this section contains more detail summary of each chapter.

Chapter 2 provides a literature review of technologies and approaches for the creation and distribution of web contents, networked games, and approaches for providing voice communication in virtual environments. The chapter first provides a classification of applications that require real-time processing of content. We then review the evolution of content distribution over the Internet, ranging from simple caching proxies to content distribution networks and personalization/localization of content. In addition, the chapter discusses current and future developments of content distribution which requires in real-time creation and distribution of content from dynamic set of dispersed sources. We review two major applications belonging to this category: *state information* processing and *communication information* processing in distributed virtual environments. For state information processing, a review of game server architectures is presented. For communication information processing, we focus on techniques and approaches for providing immersive voice communication in distributed virtual environments. Finally, the chapter reviews network and server infrastructure support for these emerging applications. In particular, we present several overlay network architectures that enable application-level routing and quality of service between processing servers and discuss the use of a shared server infrastructure for service provisioning.

Chapter 3 introduces the concept of an immersive voice communication service for multiplayer online games and describe several delivery architectures for this service. These architectures include central server, peer-to-peer, distributed locale servers, and distributed proxies. A simulation model framework is then proposed to capture game delivery scenarios including player distributions in the physical network and avatar distributions in the virtual world. Three avatar aggregation behaviours are introduced: loner, clan, and crowd. Finally, we describe key parameters used for the performance evaluation of delivery architectures: interactive delay and network bandwidth usage. Some aspects of this chapter was published in (Nguyen et al., 2004c) and (Nguyen et al., 2005a).

An investigation on the use of the central server architecture is provided in Chapter 4. We provide two optimization objectives for choosing an optimal central server from a set of potential servers. One minimizes the average delay from all participants to

the central servers which is based only on player distribution. The other minimizes the interactive delay metric, which requires both player distribution in the Internet and avatar distribution in the virtual world. The chapter also proposes a dynamic relocation of the central server in response to changes in player distribution due to time zone differences. Finally, the performance evaluation shows the advantages of the proposed solutions. This work was published in (Nguyen et al., 2004e).

Chapter 5 examines the use of the distributed locale server architecture. In this architecture, the game virtual world is partitioned into smaller areas called locales and each locale is assigned to a server. We propose an analytical model for optimizing the latency of this architecture and develop a heuristics approach based on a graph algorithm. The analytical model is based on an Integer Linear Programming (ILP) formulation which provides an exact solution to the problem but is subject to high computation complexity. In the heuristic approach, we produce a new multi-layer graph representation of the problem and devise a greedy heuristic based on this graph. The greedy heuristic has low run time complexity and is suitable for practical implementation. The simulation experiments show that the greedy heuristic solution is within 5% of optimal in all cases. The simulation results also show that the distributed locale server architecture can reduce the overall delay by around 20% compared to an optimally located central server and can have lower network bandwidth usage than the central server. Various aspects of this work were published in (Nguyen et al., 2004c), (Safaei et al., 2005), and (Nguyen et al., 2005a).

Chapter 6 examines the performance of the distributed proxy architecture. In this architecture, game players are assigned to proxy servers based on the physical location of these players with respect to these servers. We formulate the server assignment problem as an integer linear program (ILP) and propose heuristics for the problem based on the multi-layer graph model introduced earlier. The results show that the ILP model has a large run time complexity while the greedy heuristics is very efficient. Since audio streams are forwarded between proxies either using unicast or multicast, the chapter then evaluates the bandwidth cost saving of network multicast in different avatar grouping behaviours and player distribution scenarios. In addition, the effect of varying the number of proxy servers on communication delays and

network bandwidth usage is investigated. The results show that multicast is effective only when the average number of avatars in hearing ranges is large (i.e crowds). In this situation, using multicast reduces network bandwidth usage by more than 50% compared with unicast. The number of proxies also has significant impact on the latency. If only a small number of proxies are available, the use of distributed proxies result in higher delay than the central server. However, this delay can be reduced to half the delay of the central server when a large number of proxies are available. Various aspects of this research were published in (Nguyen et al., 2004d) and (Nguyen et al., 2004b), and are currently under review in (Nguyen et al., 2005b).

Chapter 7 presents an overall performance evaluation of delivery architectures. We first highlight the architectural requirements of these architectures. Then, we carry out simulation experiments to compare the performance of two distributed server architectures: distributed locale servers and distributed proxies. Based on these experiments and other simulation results from Chapter 4 to Chapter 6, we present a summary of overall performance evaluations and recommendations of delivery architectures in different game scenarios. These recommendations are given based on server resource availability, capacity of network, multicast, and game's avatar aggregation behaviors. This work was published in (Nguyen et al., 2004a) and is currently under review in (Nguyen et al., 2005b).

Chapter 8 presents a processing resource management architecture for sharing processing resources among various real-time content creation applications including the immersive audio mixing application. We first briefly describe processing requirements of an immersive audio mixing server. This is followed by a discussion in Quality of Service (QoS) provisioning issues for real-time content creation applications. Due to the inability to have a prior knowledge of processing times for scheduling, a processing resource scheduling algorithm called Start-time Weighted Fair Queueing (SWFQ) is proposed. From analysis and simulation, it is shown that SWFQ offers good fairness and delay properties compared with current schemes. In fact, the fairness of SWFQ is comparable to Weighted Fair Queueing (WFQ) and the delay behavior is better than Start-time Fair Queueing (SFQ). Various aspects of this research were published in (Nguyen et al., 2003a) and (Sabrina and Nguyen, 2005).

Chapter 9 concludes the thesis with a summary of results and recommendations. We also identify possible future work in this area.

1.3 Contributions

The contributions contained in this thesis are listed below.

- Extends existing models for distributing game states and applies these for providing an immersive voice communication service to multiplayer online games. These architectures include central server, peer-to-peer, distributed locale servers and distributed proxies (Sections 3.5).
- Develops a simulation environment that creates geographic distribution of game players in the Internet and different avatar aggregation behaviours in the virtual world (Section 3.6).
- Provides two optimization objectives for the central server architecture and a mechanism to relocate the central server in response to changes in player distribution due to time zone differences (Chapter 4).
- Develops an analytical model and a heuristics for the server assignment problem in the distributed locale servers. The analytical model is based on an integer linear programming formulation which provides optimal solution but is subject to high computation complexity (Section 5.2.2). Hence, we produce a new multi-layer graph representation for the problem and devise a novel greedy heuristic algorithm based on this graph (Section 5.2.3). The efficiency of the greedy heuristic is demonstrated in simulation experiments. In addition, the effect of varying the number of servers and avatar aggregation behavior is also investigated (Section 5.3).
- Develops a mathematical formulation for the server assignment problem in the proxy architecture. A heuristics for the problem is also provided and evaluated. In addition, we examine the performance of the distributed proxy architecture with respect to two key issues. First is the efficiency of multicast in reducing

bandwidth usage in different avatar grouping behaviours and player distribution. Second is the effect of varying the number of proxies on the interactive delay and network bandwidth usage (Chapter 6).

- An overall performance evaluation of different delivery architectures is investigated. We focus on providing a performance comparison of distributed locale servers and distributed proxies. We also provide a performance summary and recommendations on the applicability of these architectures in different game delivery scenarios (Chapter 7).

- Designs a resource management mechanism for sharing processing resources among various real-time content creation applications such as media transcoding, encryption, and the immersive audio mixing application. In particular, we propose a novel processing resource scheduling algorithm called Start-time Weighted Fair Queuing which offers good fairness and better delay behaviour than the current algorithm in the literature (Chapter 9).

1.4 Publications based on Thesis

C. D. Nguyen, F. Safaei, P. Boustead, "Optimal assignment of distributed servers to virtual partitions for the provision of immersive voice communication in massively multiplayer games," To appear in *Special Issue in Computer Communications Journal, Elsevier*.

F. Safaei, P. Boustead, C. D. Nguyen, J. Brun, M. Dowlatshahi, "Latency driven distribution: infrastructure needs for participatory entertainment applications," in *IEEE Communication Magazine on Entertainment Everywhere: System and Networking Issues in Emerging Network-Centric Entertainment Systems*, vol. 43, no. 5, May, 2005, pp. 106-112.

F. Sabrina, C. D. Nguyen, S. Jha, D. Platt, F. Safaei, "Processing resource scheduling in programmable nodes," in *Journal of Computer Communications, Special Issue on Activated and Programmable Internet*, vol. 28, no. 6, April 2005, Elsevier, pp. 676-687.

C. D. Nguyen, F. Safaei, P. Boustead, "Comparison of distributed server architectures in providing immersive audio communication to massively multi-player online games," in *Proceedings of Australian Telecommunications Network and Applications Conference (ATNAC)*, Sydney, Australia, December 2004, pp. 499-505, ISBN: 0-646-44190-6.

C. D. Nguyen, F. Safaei, P. Boustead, "Performance evaluation of a proxy system for providing immersive audio communication to massively multi-player games," in *Proceedings of 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), Globecom 2004*, Dallas, TX, USA, November 2004, pp. 192-199.

C. D. Nguyen, F. Safaei, P. Boustead, "A distributed server architecture for providing immersive audio communication to massively multi-player online games," in *Proceedings of IEEE International Conference on Networks (ICON)*, Singapore, November 2004, pp. 170-176.

C. D. Nguyen, F. Safaei, P. Boustead, "A distributed proxy system for provisioning immersive audio communication to massively multi-player games," in *Proceedings of ACM SIGCOMM Workshop on Network and Systems Support for Games (Netgames)*, Portland, Oregon, USA, August 2004, p. 166.

C. D. Nguyen, F. Safaei, D. Platt, "On the provision of immersive audio communication to massively multi-player online games," in *Proceedings of the ninth IEEE Symposium on Computers and Communications (ISCC)*, Alexandria, Egypt, June 28th - July 1st 2004, pp. 1000-1005.

C. D. Nguyen, D. Platt, F. Safaei, "Design of processing resources scheduling in programmable networks," in *Proceedings of Australian Telecommunications Network and Applications Conference (ATNAC)*, Melbourne, Australia, December 2003, ISBN: 0-646-42229-4.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews the current technologies and approaches for the creation and distribution of web and multimedia content. We focus on real-time creation and distribution of interactive multimedia content and the infrastructure support for delivering these contents.

In early web applications, the delivery of content is simply about retrieval of hypertext documents from a web server. As Internet applications evolve, the nature of delivering content is also changing significantly. In many current applications, content is often processed or customized in *real-time* by a server or a proxy before being delivered to end users. This content includes web content, multimedia, state information and communication information in distributed virtual environments. Applications that require *real-time creation of content* are classified as follows:

- *Content personalization/localization:* Advanced content service delivery enables personalization of content by using application proxies located at network edges. Example of these services include language translation, insertion of regional data and personalization and customization of web pages.
- *Content transformation:* This class of application includes transcoding, encryption/decryption and compression/decompression. Typically, a server or a

proxy is placed between a content source and a destination in order to seamlessly provide application specific processing on the packet flow in real-time.

- *Real-time creation of content from dynamic sets of dispersed sources:* Applications belonged to this class include state information and communication processing in multiplayer online games and distributed virtual environments. For example, a game server dynamically processes packets carried game state information, generates new game state and sends back to game clients. An audio mixing server mixes voice streams from players and sends the mixed streams to other players within their hearing range.

The chapter is organized as follows. Section 2.2 reviews the evolution of web content distribution. In Section 2.3, the thesis describes the current and future applications that require real-time creation and distribution of content from a large number of disperse sources. These include state information and communication processing in multiplayer online games and distributed virtual environments. In Section 2.4, the thesis reviews the server infrastructure support for these applications. Section 2.5 concludes and points out major research issues that will be considered in the thesis.

2.2 Evolution of Content Delivery

This section reviews the evolution of web content delivery over the Internet. Due to the rapid growth of the Internet and the world wide web, several approaches have been developed for efficient delivery of web contents. These approaches range from simple proxy caching to web cluster servers and content distribution networks. The objectives of these approaches are to improve the scalability of content delivery and reduce the response time. Finally, the section introduces the concept of content service networks (Ma et al., 2001) and several advanced techniques for enabling personalization/localization of content.

2.2.1 Proxy Caching

One of the earliest techniques to enhance the performance of web delivery is proxy caching. Proxies are originally designed to allow network administrators to control Internet access from within an intranet. Moreover, proxies have been used widely for caching web documents. Cache proxies serve as repositories for frequently requested document, hence, reduce network traffic and response time (Mahanti et al., 2000). A cache proxy receives Hypertext Transfer Protocol (HTTP) requests from clients. The proxy returns the cache object if the requested object is found.

The simplest caching technique for web content is the client web browser cache which is only useful for a given user. This is the first-level caching, while caching at a proxy is the second level cache. There are three basic approaches for proxy cache deployment as presented in (Barish and Obraczka, 2000):

- *Consumer-oriented:* In this case, the Proxy cache is deployed near consumers. Since this proxy is subject to single point of failure, there are two approaches for alleviating this problem. The first approach is auto-configuration of browser based on IETF Web Proxy Auto-Discovery Protocol (WPAD) (Gauthier et al., 1999) for locating nearby proxy caches. The second approach is transparent proxy, which intercept HTTP requests at network edges (e.g., router or layer 4 switch) and redirect these requests to suitable web cache servers. This approach eliminates the need for browser configuration but violate end-to-end argument by not maintaining constant end-points for the flow (Gauthier et al., 1999).

- *Provider-oriented:* Caches are positioned or maintained by the content provider in order to improve access to a logical set of content such as audio and video. For example, in the reverse proxy caching, caches are deployed near the origin of content instead of near clients.

- *At strategic points in the network:* An example of this strategy is adaptive web caching, which views caching problem as one of optimizing global data dissemination. This scheme aims to deal with “hot spot” in the network, where

various short-lived Internet contents can become massively popular and in high demand.

2.2.2 Web Server Cluster

To cope with a rapid increase in the number of users accessing popular web sites, there is a need for faster and scalable web servers. One solution is to use a pool of servers tied together to act as a single unit called a *server cluster* (Schroeder et al., 2000). A server cluster provides a better cost/performance and a simpler path for capacity extension compared with one large server.

Most of web server clusters are transparent to client browsers. Reviews of web-cluster systems are presented in (Cardellini et al., 2002) (Bryhni et al., 2000) and references therein. These systems consist of multiple server nodes distributed in a local area and employ mechanisms to spread client requests among these nodes. Various request dispatching algorithms and internal routing mechanisms have been proposed to design and implement scalable web-server systems. The aims of these algorithms are to achieve high throughput, load balancing, and low response times at the web server cluster. There are three main approaches for distributing client request to multiple servers:

- *DNS-based approach:* An authoritative Domain Name Server (DNS) returns different IP addresses for the same domain name based on a list of servers corresponding to this web site. For load balancing purpose, requests are usually assigned to servers in a round robin fashion.
- *Server-based approach:* This approach use HTTP redirection feature in accordance with DNS based distribution of requests. DNS is first used to distribute requests among a set of servers. If a request reaches an overloaded server, instead of returning the page, the server responds with a HTTP message to direct the client to a new server.
- *Network-based approach:* In this approach, a specialized node is deployed to distribute HTTP request among servers in the cluster. In different system implementations, this node is called by various names such as HTTP dispatcher,

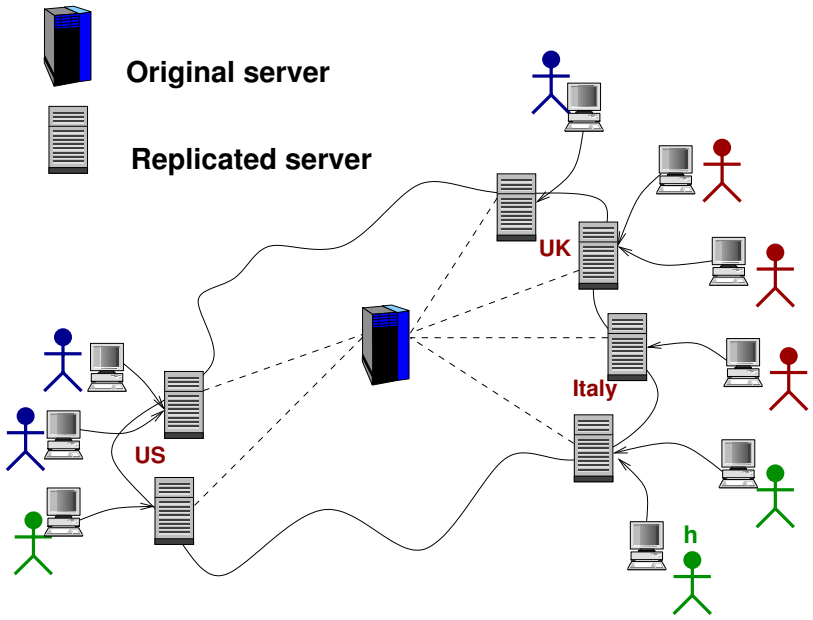


Figure 2.1 A content distribution network.

web switch, application layer switch, or layer 4-7 switch. The switch remaps incoming HTTP requests at layer 3 or layer 2 based on information at layer 3 and 4 or layer 3 up to 7. As presented in (Schroeder et al., 2000), these combinations result in L4/2 or L4/3 (layer 4 switching with layer 2 or 3 packet forwarding) and L7/2 or L7/3 (layer 7 switching with layer 2 or 3 packet forwarding).

2.2.3 Content Distribution Networks

The next advance in content distribution is to replicate web contents in various servers or clusters in different geographical locations that are close to clients in order to improve request latency and throughput. This solution is called a Content Distribution Network (CDN), as shown in Fig. 2.1. Currently, popular commercial CDNs include Akamai [1], Exodus [2] and Digital Island [3]. Akamai comprises 15,000 servers

[1] Akamai, http://www.akamai.com
[2] Exodus, http://www.exodus.com
[3] Digital Island, http://www.digitalisland.com

in 1100 networks and more than 65 countries. It is claimed by Akamai that 15% of total web traffic in the Internet is from Akamai network. In addition to enhancing the scalability of web hosting, Akamai also enables information and applications to be delivered from locations close to the customers.

The traditional approach (without CDNs) is to use a web server cluster for improving the performance scalability and rely on caching proxies for reducing the number of hits. This approach has some limitations:

- If the web server cluster is designed for peak load, the infrastructure cost will be high and utilization is likely to be low or if the cluster is designed for average load, the performance will be poor during peak periods.
- If the network between the web server and customers is congested, the users will suffer from poor performance.
- The web server cluster will not be able to cope with unexpected large number of requests referred to as "flash crowds". The most famous example is the 1998 distribution of the Monica Lewinsky Starr report, which was successfully delivered to 20 million Americans over a short period of time (Douglis and Kaashoek, 2001). The first server contained the document was swamped with requests until other sites that had successfully downloaded the document made their own copies to spread the load.

The Starr report example demonstrated the advantage of a content distribution network in response to flash crowds. By using CDNs, the service can automatically scale without continually increasing the size of the web cluster server. In addition, while web clusters can only reduce response times at the original server site, CDN can improve latency and reliability of content delivery and is resilient to congestion in the core of the Internet due to smaller distance from customers to replicated servers. Also, security mechanisms are employed to ensure the content servers are protected from attacks, thus ensuring performance stability. In addition to traditional web contents, CDN is also suitable for value-added services such as streaming media over the Internet (Cranor et al., 2001).

The main components of a CDN include a content delivery infrastructure and a request routing infrastructure (Green et al., 2000). The content delivery infrastructure is a set of "surrogate servers" located at the edge of the network. The request routing infrastructure diverts user requests to suitable surrogate servers.

The CDN surrogate servers may either replicate and deliver only selected objects within the HTML document or the whole web site. A surrogate server also provides dynamic content assembly, localization and personalization of content, and streaming media on demand. In particular, dynamic content assembly is supported by using a standard called Edge Side Includes (ESI) [4] and localization and personalization of content is enabled by another standard called Internet Content Adaptation Protocol (ICAP) (Elson and Cerpa, 2003).

CDNs employ some request-routing mechanisms for efficiently redirecting user requests to appropriate CDN servers to reduce latency and balance load. Common approaches include HTTP redirection, DNS redirection, and network layer. In the case of HTTP redirection, clients are redirected to an optimal replica server via the use of HTTP protocol response codes (Fielding et al., 1999). A client establishes HTTP communication with one of the replica servers. The initially contacted replica server can then either choose to accept the service or redirect the client again. In DNS redirection (Cooper et al., 2001), a more sophisticated client to replica communications is enabled by using enhanced DNS servers. When a client resolves the name of an origin server, an enhanced DNS server sorts the available IP addresses of the replica servers starting with the most suitable replica and ending with the least suitable replica based on some metrics. In the case of network layer redirection (Agarwal et al., 2001), anycast addresses are used for directing requests to replicated servers.

2.2.4 Advanced Content Service Delivery

The previous sections introduce several approaches for content *distribution*. This section describes advanced techniques for enabling *processing* of content at surrogate servers or proxies in order to provide richer services. There are two major standards: Edge Side Includes (ESI) for dynamic content assembly at edge servers and Internet

[4] Edge Side Includes, http://www.esi.org/

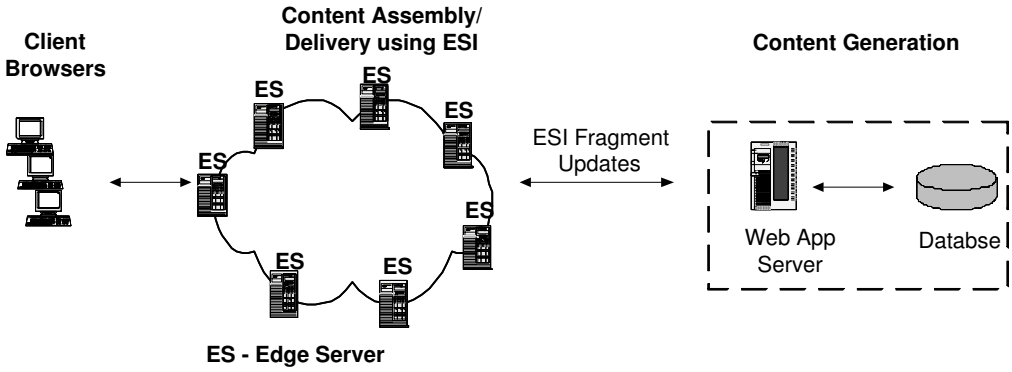


Figure 2.2 Content delivery/assembly using Edge Side Includes (ESI).

Content Adaptation Protocol (ICAP) for localization and personalization of content. In addition, a recent content delivery concept called Content Service Networks is also introduced.

Edge Side Includes (ESI) [5] defines a simple markup language to facilitate the dynamic content creation at edge servers using cached objects at these servers and dynamic components that are retrieved from the origin server. As indicated in Fig. 2.2, ESI separates content delivery from content generation, thus lowering the need to retrieve the complete pages and substantially reducing the load at content generation infrastructure. To enable this, ESI defines a template which consists of ESI fragments and their expiration times (or Time To Live TTL) as shown in Fig. 2.3. The fragments are individually obtained from the origin server after TTL expiration.

Internet Content Adaptation Protocol (ICAP) (Elson and Cerpa, 2003) is a lightweight protocol for adapting HTTP requests and responses that enable content localization and adaptation in CDNs. Examples include formatting HTML for display on special devices, language translation, and localized advertisement. Fig. 2.4 gives an example of an ICAP language translation service operation.

1. A client requests an object from the origin server through an ICAP-client surrogate.

[5] Edge Side Includes (ESI) Overview, http://www.esi.org/overview.html

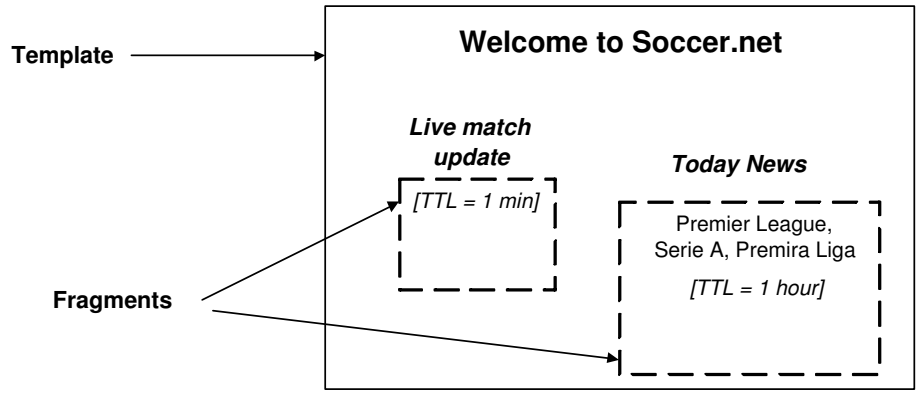


Figure 2.3 Example of an ESI template consisting of ESI fragments and their expiration times.

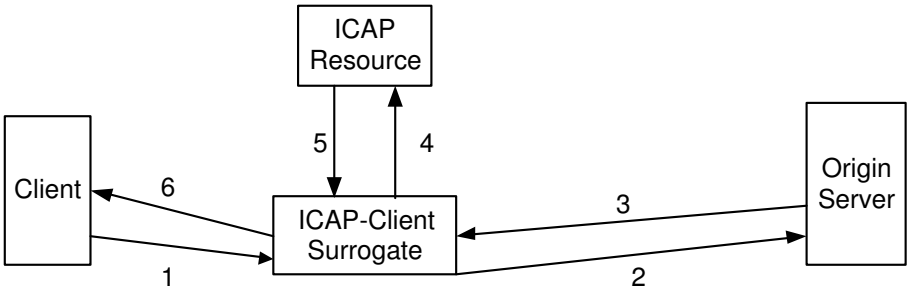


Figure 2.4 Example of ICAP operation.

2. The surrogate forwards the request unchanged to the origin server.

3. The origin server responds with the partially complete HTML document which contains a link to the ICAP language translation service.

4. The ICAP client intercepts the server's reply and sends a request to the ICAP resource server to obtain the translation service.

5. The final HTML document is assembled at the ICAP-client.

6. The complete translated HTML document is forwarded to the client.

While ICAP protocol enables a limited range of content localization and adaptation

service in current CDN, a new concept called Content Service Network (CSN) (Ma et al., 2001) supports a wider range of content processing capabilities. CSN can be viewed as an overlay network of application proxies that focuses on providing content processing in stead of storage and caching as in current CDNs. CSN provides a range value-added services including web content localization and personalization as well as other applications such as streaming media transcoding. CSN aims to provide these services for a wide range of customers including content providers, end users, ISPs, and CDNs.

2.3 Future Development of Content Creation and Delivery

In the previous section, we reviewed the evolution of web content distribution up to now. The convergence of the Internet and entertainment is likely to lead to the proliferation of interactive entertainment applications and challenges for delivering *interactive entertainment content*. This type of content includes *state information* in distributed virtual environments and *communication information*, such as voice streams, for enhancing communication in these environments. These contents are often created and distributed in real-time from a dynamic set of dispersed participants.

In this section, we describe the current development of multiplayer online games, which is the most popular application in "state information processing" category. In addition, we outline voice support systems in current multiplayer games and review research results for enhancing this service. In particular, we describe technologies and approaches for providing *immersive* voice communication to virtual environment.

2.3.1 Multiplayer Online Games

Multiplayer online games have become very attractive applications in the Internet. Some estimates indicate that by 2009 more than 230 million people will be playing

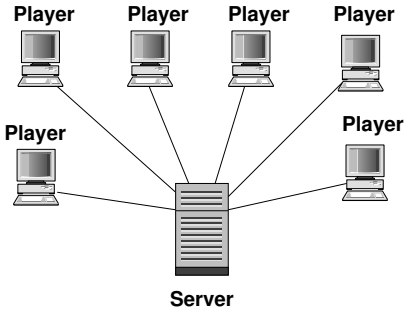
multiplayer online games [6]. Most current commercial multiplayer online games are based on the client-server model. They can fall into two categories. In the first category, one peer can act as a server or a stand-alone server is used. In either of these cases, this server can only support a small number of players (typically well under 100). In the second category, a central server in a form of a dedicated server or a server farm is used. This type of server can support from 1000 to more than 100,000 players. This category is also called Massive Multiplayer Online Game (MMOG).

Massive multiplayer online games (MMOG) have become very popular applications recently. A report (Aas et al., 2003) shows that the revenue for MMOG is estimated at $635 million in 2003, and by 2005 should reach $1.8 billion. Another report by Zona Inc. [7] shows that the number of MMOG subscribers in 2002 is about 1.6 millions in US alone, and will reach 10.9 million in 2006.

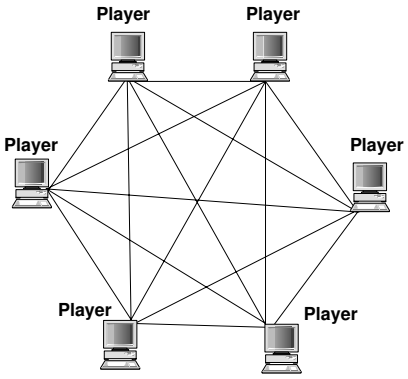
The number of participants in a single MMOG is very high and tends to increase when the game becomes more popular. However, most of current commercial game servers can only support few thousands players. For example, although EverQuest [8] claims to support over 100,000 players simultaneously, there are about 50 servers and each server can only handle about 2000 players. Since each server hosts a separate virtual world, Everquest actually supports 50 instances of the game, not a single game with a large virtual world.

There are various works on network and server support to enhance the scalability and latency of MMOG. To address the scalability problem, the industry has been developing scalable server solutions for hosting MMOG. Notable examples are Terazona system developed by Zona [9] and OpenSkies system developed by Cybernet Systems[10]. These systems are built based on a cluster of servers. Another notable system is Butterfly Grid [11], which uses grid computing to support a large number of users.

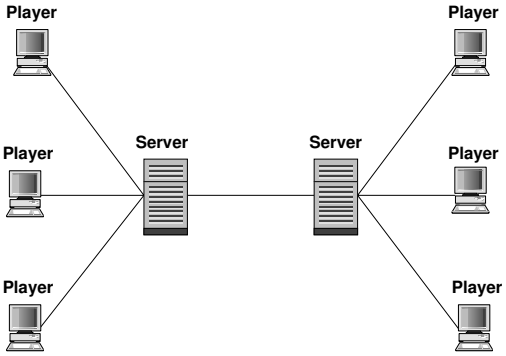
[6]DFC Intelligence, "*The Online Game Market 2004*", August 2004, http://www.dfcint.com
[7]Zona Inc, http://www.zona.net
[8]EverQuest, http://www.everquest.com
[9]Terazona System, http://www.zona.net
[10]OpenSkies System, http://www.openskies.net
[11]Butterfly Grid, http://www.emergentgametech.com



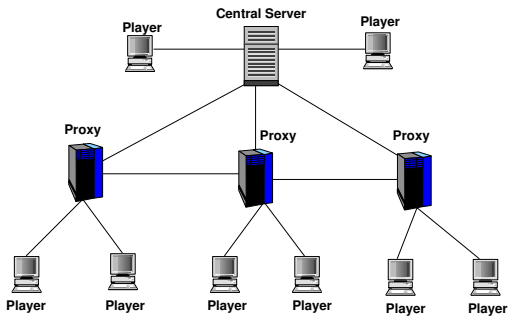
(a) Client-server



(b) Peer-to-peer



(c) Distributed servers



(d) Proxy

Figure 2.5 Game server architectures.

This section outlines existing multiplayer online game architectures.

2.3.1.1 Client-server

In the client-server architecture (Fig. 2.5a), the central server processes all state information. When each player moves or does a certain game action, this command is sent to the central server. The server collects all game commands from clients, updates game state and sends these updates back to each client. The client machine renders the graphic output and sound based on the new game states. This architecture is common in most commercial networked games. In the centralized server architecture, it is easy to implement functionalities such as security and accounting. In addition, since all game states are updated at the centralized server, global consistency is always maintained. However, this architecture poses a scalability problem and the central server is a single point of failure. Also, additional delay is introduced since interaction between players needs to be updated via a central server.

2.3.1.2 Peer-to-peer

Currently, the peer-to-peer architecture (Fig. 2.5b) is used in some real-time strategy games where the number of participants is small. Each game client sends its game events to all other game clients and receives events from other clients. This approach limits the scalability of the game due to large amount of game state information each player need to send/receive. Hence, it is typically used for some real-time games with a small number of participants or when participants are on a LAN. In addition, due to direct peer-to-peer communication, this architecture is subject to cheating and security problems. The advantage of peer-to-peer architecture is lower communication delay between game players compared to the client-server approach.

2.3.1.3 Distributed Servers

To increase the scalability of game servers distributed server architectures (Fig. 2.5c) have been proposed in the literature in (Barrus et al., 1996) (Cronin et al., 2001) and references therein. There are two ways for distributing game state information processing: *virtual location based partitioning* and *physical location based partitioning*. These architectures can reduce the scalability problem or, in some cases, reduce the

network latency compared with the central server architecture. However, there is a need to maintain game state consistency between servers.

In virtual location based partitioning, the game virtual world is partitioned into smaller areas called locales or zones, each area is assigned to a server. This approach is proposed in (Barrus et al., 1996) and has been used for distributing game state information among servers in the Butterfly Grid. This approach mainly aims to increase the scalability (in terms of number of players) as servers are typically located in the same physical location such as a server farm. In this approach, the game state consistency needs to be maintained when avatars move across the border of two adjacent locales, which is assigned to two different servers.

In physical location based partitioning, distribution of servers are based on the physical location of servers with respect to clients. Game clients are connected to a nearby server as in the client-server architecture. Each server keeps its own copy of game states and exchanges messages with other servers using a peer-to-peer architecture. This architecture is also called a mirrored server architecture (Cronin et al., 2001), which is a hybrid between the client-server and the peer-to-peer architecture. A key issue of this architecture is to maintain game state consistency between servers. Several synchronization techniques (Cronin et al., 2002a) has been proposed to minimize the inconsistency between these servers and ensure the accurate performance of the game. By an efficient assignment of players to close servers, this architecture could achieve smaller latency than the client-server architecture.

2.3.1.4 Proxy-based Architecture

The work in (Mauve et al., 2002) proposes a proxy system to avoid the bottleneck of the client-server architecture and limitation of the peer-to-peer architecture. As shown in Fig. 2.5d, a player is either connected directly to the central server or via a nearby proxy. The central server trusts these proxies and delegates some functionality to these proxies. These proxies form an overlay network. Hence, this architecture has some of the advantages of overlay networks such as congestion control by routing traffic around the congested areas, fault detection and rerouting. Also, since proxies are close to players, latency is improved. Proxies also prevent cheating and ensure

Figure 2.6 Example of a game virtual world

fairness between players.

2.3.2 Group Communications in Multiplayer Online Games

In most current multiplayer online games, players can communicate only by typing a text message which will appear on the computer screen of other players. Other players can respond to this by typing a reply message. This is very unnatural and slow. In some fast-paced games, typing an alert message may be too slow and other players may get killed before getting the alert. In addition, typing text would interfere with players' control of their characters' movements. Voice would allow participants to play and communicate with others at the same time.

A multiplayer online game simulates a virtual world, where each game player is represented as an avatar. Fig. 2.6 shows an example of a virtual world which consists of several avatars. These avatars communicate to exchange information and do various actions. Since networked games are becoming more realistic, there are more interac-

tions between players. As a result, communications play a vital role. For example, in First Person Shooter (FPS) games, members in a team of soldiers need to talk about tactics for an attack, give instructions for using weapons, and alert others about the approaching enemy.

To enhance the game experience, inter-person communication is a key issue. Currently, there is a strong push by game service providers to add voice communications to networked games. Voice communication has been implemented in some console-based and PC-based games. Most of these voice communications are implemented in a party-line fashion and have certain limitations. These systems' basic features are described in the next section.

2.3.2.1 Voice Communication Systems in Current Multiplayer Online Games

One of the common voice support systems in current networked games is the Xbox Live [12] which was released by Microsoft in 2002. Voice communications in most of Xbox games are typically implemented as a party line fashion. For example, in a FPS game, there is only one voice channel for the whole game (in all against all play) or one voice channel for each team (team-based game). A study in (Hew et al., 2004) shows that the Xbox voice communication system lacks the ability to control what is heard and sent through the voice channel. A number of participants felt that various conversations in the voice channel were not intended for them. Due to only one voice channel, they can hear incoherent speech from other players that are not in their proximity and not of interest to them. In addition, they could neither control nor determine who was listening to their transmission. In some games such as MotoGP and Halo 2, Xbox Live also supports limited proximity-based voice communication, allowing players to talk to each other if they are close to each other. In this situation a game context, such as avatar position, is used, however, no spatial voice rendering and distance based voice attenuation is implemented.

Voice has been implemented in some PC-based games such as Counter Strike [13]. Similar to Xbox live, the voice channel is implemented as a party line. However,

[12] http://www.xbox.com/en-US/live/about/features-voice.htm
[13] http://www.counter-strike.net/

players can call up a scoreboard menu and use mouse to choose to listen or mute other players. This personalized option enables a player to have more control of who is being heard. Also, there is some visual cue as avatar lips move when a player talks. This type of conversation would be suitable for games, in which the number of participants is small and players only need to speak to their team. In a game that has a large number of participants, this voice communication scheme may not be suitable as each participant may interact with many different players in a game session. This scheme requires explicit control of the voice channel from the users. Similar to Xbox voice communication, no game context is used and the voice is not spatially placed nor attenuated.

SideWinder Game Voice [14] is a supplemental system that provides voice communications between players in PC-based games, regardless of the types of game they play. This system was released by Microsoft in August, 2000. The system consists of a special headset, a control pad and a software component that is integrated with game player's session. The system is game-independent and relies mainly on MSN Instant Messenger voice chat technology. There are four voice channels, and each channel is typically assigned to a player. A player can choose any on and off combination of these channels by using buttons on the control pad. All player usernames will appear in a chat room list and each player can explicitly choose who to talk to. This is similar to common chat communication. There is no linkage between game context such as player position and team allocation with the voice support function.

In terms of delivery architectures, most current systems use peer-to-peer, which is direct voice transport connection between users. This is similar to voice chat communication in current common Instant Messenger such as MSN Messenger or Yahoo Messenger. A voice stream from each user needs to be sent to the rest of users in the same party-line or chat room. UDP and RTP are typically used for transporting a voice stream. Hence, the number of sending voice streams is equal to the number of listeners. Due to limited user access bandwidth, these systems only support a small number of simultaneous speakers.

In short, most voice communication systems in current multiplayer online game are

[14] http://www.microsoft.com/presspass/features/2000/aug00/08-24gamevoice.mspx

designed for FPS or racing games with small number of users. Due to the party line fashion of voice delivery, each player can only talk to a few people such as members of the same team. In addition, the voice delivery is not based on spatial location of a speaker with respect to the listener and the distance between them. In the next section, we will describe research projects that provide more realistic voice communication in distributed virtual environments. Since a multiplayer online game is a particular class of distributed virtual environments, the techniques used in distributed virtual environment are generally applicable to multiplayer online games.

2.3.3 Immersive Voice Communication in Distributed Virtual Environments

This section describes various techniques and approaches in providing *immersive* voice communication in Distributed Virtual Environments (DVE). These approaches enable spatial voice communication between participants and enrich the experience of the virtual environment. Each system design addresses different issues such as network bandwidth limitation for the voice delivery or processing resource scalability for performing audio mixing operations.

To provide *immersive* voice communication, mixing of audio streams is often required. We refer to this operation as *communication information processing*. In addition to "state information processing", "communication information processing" in distributed virtual environments has recently received considerable attention from the research community.

Early research work on voice support for DVE emerges in the mid 1990s. At this time, the importance of spatial audio communication alongside the 3D display of a virtual environment is already realized. A study in (Hendrix and Barfield, 1996) shows that addition of spatial sound significantly increases the sense of presence in the virtual environment. Notable systems that support voice communications at this time are Distributed Interactive Virtual Environment (DIVE) (Carlsson and Hagsand, 1993) and Scalable Platform for Interactive Environment (Spline) (Anderson et al., 1995) (Waters et al., 1997). Although these papers have mentioned the significance of spatial voice communications, the system design, implementation and various is-

sues related to provisioning spatial voice communications are not adequately addressed.

DIVE is one of the first DVE system that provides virtual voice conference for participants. DIVE is based on peer-to-peer approach, where peers communicate by using IP multicast. To address the lack of IP multicast support in many carriers, a latter enhancement of DIVE (Frecon et al., 1999) presents an architecture called DiveBone that uses a proxy server as an application-level multicast node. These servers form an application-level multicast backbone. The DiveBone also provides visual analysis of the connection architecture and network traffic for remote maintenance operation.

In Spline framework, the importance of voice immersion is addressed although the specific design for spatial voice communications between users is not provided. Spline uses peer-to-peer voice communication between users. If an user has adequate access bandwidth, the audio mixing is done at the client computer. Otherwise, users are connected to an access server that interfaces with the main system and provides customized mixing for each connected user.

Several research projects in this area focus on mechanisms to support voice in DVE through heterogenous networks such as the Internet. One example is the work in (Radenkovic et al., 2002) (Radenkovic and Greenhalgh, 2002) that provides an audio service for Collaborative Virtual Environments (CVEs). A technique called Distributed Partial Mixing (DPM) is used to dynamically adapt to varying numbers of speakers and network congestion. This scheme supports optimization of bandwidth utilization across multiple related audio streams while maintaining fairness to TCP traffic in best effort networks. For example, when there is congestion, performing mixing (e.g., in a server or proxy) would reduce the number of peer-to-peer audio flows in the network. In addition, the deployment of DPM is based on application layer multicast in which a mixing server or a client computer can be a multicast node. The multicast trees are self-organized when clients join or leave or when network condition changes (e.g., in response to congestion). Issues of deploying DPM server over wide-area networks such as DPM placement, reconfiguration is also discussed. However, this work is mainly concerned with optimizing bandwidth, and minimizing latency for voice communication has not been considered.

The work in (Greenhalgh and Benford, 1999) discusses the design of CVEs that scale to large numbers of participants yet still afford spatial voice communication. The scalability is achieved by performing operation and summarization based on third-party objects (e.g., structuring regions, aggregate views, and dynamic crowds of participants). The system implementation, namely Massive-2, is based on a dynamic and self-configuring hierarchy of multicast groups. The paper states that the combination of third-party objects and multicast would improve the scalability of the architecture including voice stream delivery.

The research in (Bolot and Parisis, 1998) examines issues related to adding voice to networked games such as the MiMaze game. The paper discusses all stages of voice communication architecture, including voice generation, voice transport, and voice restitution. For voice generation, echo cancellation technique is discussed to prevent the voice of remote player from being captured by the microphone. For voice transport, the paper considers peer-to-peer voice delivery between users by using IP multicast, UDP and RTP. For voice restitution at a listener, the paper discusses some 3D audio processing techniques and synchronization of visual and audio sources. This research only concentrate on a peer-to-peer architecture while the use of other delivery architectures is not addressed.

2.3.3.1 Spatial Audio Rendering Techniques

To provide immersive voice communications in DVE, spatial audio rendering techniques need to be used. These techniques render the sound in a 3D environment based on the spatial location of the sound sources with respect to the listener in the virtual world.

For many applications, accurate spatial reproduction of sound can significantly enhance the visualization of 3D information. The human ear-brain interface is capable of identifying sounds in a 3D environment with remarkable accuracy. Sound perception is based on a multiplicity of cues that include level and time differences and direction-dependent frequency-response effects caused by sound reflection in the outer ear, head and torso, cumulatively referred as the head-related transfer function (HRTF) (Kyriakakis, 1998).

Review of spatial audio rendering techniques is discussed in (Kyriakakis et al., 1999) (Kyriakakis, 1998) and references therein. There are two common methods for 3D audio rendering that can be classified as "head related" transfer function and "non-head related", based on loudspeaker reproduction.

Nonhead related methods typically use multiple loudspeakers to reproduce multiple matrixed or discrete channels to convey precisely localized sound images. This technique is also called amplitude panning which models a sound field with different intensities according to the direction of the virtual source. 3D localization can be achieved if there are enough speakers available (Pulkki, 1999).

Head-related transfer function (Gardner and Martin, 1994) models the spectral and timing transformation of a sound source in free-field to the ears of the human listener, caused by the diffractions of sound by the torso and head. Since binaural localization synthesizes sound sources at the two ears of the listener, this technique is optimized for playback on the two channels of the headphones. Using this approach with two loudspeakers is possible but it requires careful crosstalk-cancellation to achieve good localization precision (Mouchtaris et al., 2000).

2.3.3.2 Dense Immersive Communication Environment (DICE)

Researchers in our research group have developed a system design called Dense Immersive Communication Environment (DICE) (Boustead et al., 2005). The project aims to design a scalable system that delivers immersive voice communication in crowded virtual spaces. A testbed has been implemented to provide this service for several popular networked games.

The basic functional elements of DICE is shown in Fig. 2.7. There are two main components: audio clients and a Scene Creation Server (SCS). An audio client installed in each player's computer captures voice (from microphone) and sends a single mono audio stream to the SCS. The audio client receives up to a fixed number (k) of mono streams with associated positional meta-data from the SCS. This meta-data is used to spatially place each of the received mono-audio streams to create an audio scene that corresponds, as accurately as possible, to the visual component of the virtual

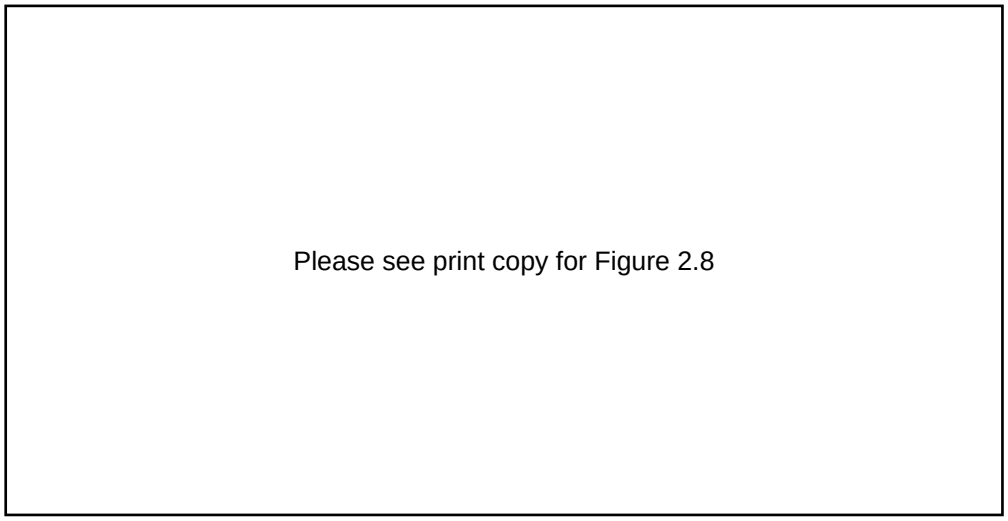


Figure 2.7 Basic functional elements of DICE, adapted from (Boustead et al., 2005).

world. If the number of avatars in the participant's hearing range is smaller than *k*, the SCS merely forwards these n streams to the participant. The SCS obtains avatar information such as position and orientation from the game server. This information is used to build a map of avatar coordinates that is used by a control algorithm to calculate mixing operations on audio streams. For example, the distance between a speaker and a listener determines the weight applied to the audio stream from that speaker, which consequently affects the volume of that speaker's voice in the mixed audio scene.

Since client access bandwidth is limited and the state information exchanges consume a large portion of this bandwidth, *k* will be small in practice. As shown in Fig. 2.8, DICE uses an angular clustering approach that groups avatars into *k* separate clusters and perform a spatial audio mixing operation on each of these clusters (in the testbed implementation, *k* is equal to 4). The server will also calculate the centre of activity of each cluster, which is the location of an imaginary audio source of the cluster mix. This information is then sent to the client and the client will render the audio scene by spatially placing each mixed audio at its centre of activity. By using intelligent methods for clustering and calculation of centre of activity, giving higher weights to nearby avatars, it is possible to construct perceptually accurate audio scenes even when the number of allowed voice streams is small.

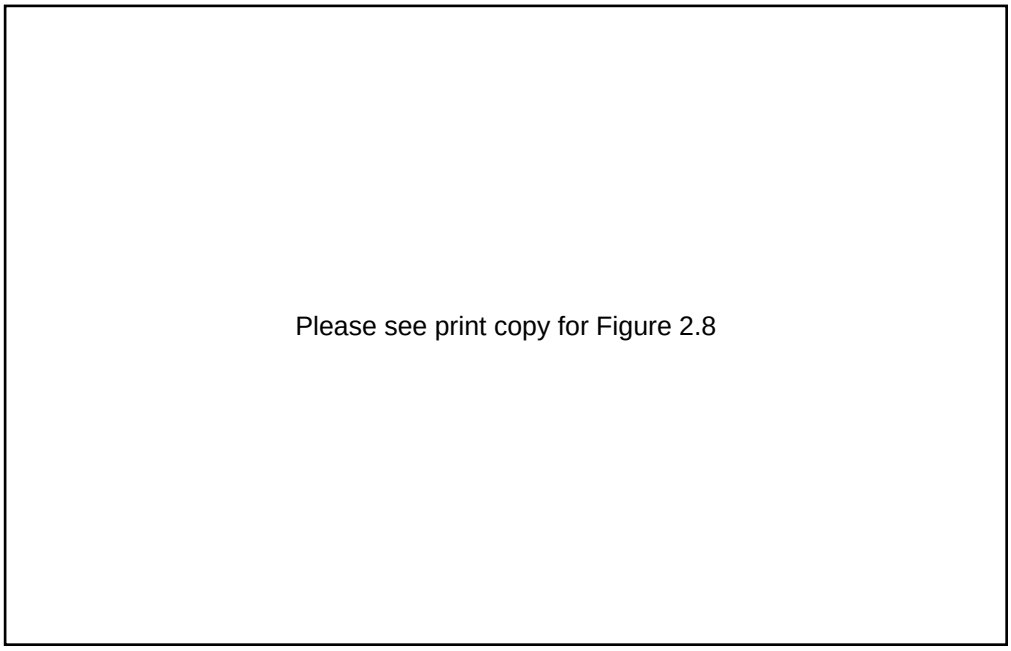


Figure 2.8 Angular clustering, adapted from (Boustead et al., 2005).

2.3.4 Audio Codec Technologies

This section introduces audio coding technologies for multimedia streaming and storage, and voice transmission over the Internet. We focus on several types of codecs which are applicable for the transmission of voices from participants in the virtual environment.

Key parameters that determine the characteristics of audio codecs are: sampling frequency, bits per sample, data rate, and processing delay. A basic audio coding scheme is Pulse Code Modulation (PCM), which is a digital representation of analog signal, where the magnitude of the signal is sampled at a uniform interval and then quantized to binary code. Using this coding scheme, uncompressed mono voice stream is sampled at 8000 samples/sec with 8 bits/sample, resulting 64kbps data rate. CD audio quality is sampled at 44100 samples/sec with 16 bits/sample, resulting in 1411200 bits/sec or about 1.5 Mbps (Harrington and Cassidy, 1999). Audio codecs attempt to relatively achieve the quality of uncompressed audio with lower data rates.

There are various audio codecs for multimedia streaming and storage which are not particularly designed for speech but audio in general. Examples are various audio

codecs that are specified in several standards developed by the Moving Picture Experts Group (MPEG) audio committee [15]: MPEG1 in 1993, MPEG2 in 1994, MPEG3 in 1998, and MPEG4 in 1999. These codecs have data rates ranging from 6kbps for low quality audio to 700kbps for CD quality audio. In addition, there are also proprietary audio codecs developed outside the MPEG standardization framework, such as Windows Media Audio (WMA) [16] from Microsoft and Real Audio (RA) [17] from Real Networks.

These above audio codecs do not take advantage of statistical properties of human speech for coding and are not suitable for voice transmission. The following sections review several audio codecs used for speech compression and transmission over the IP network.

2.3.4.1 Speech Codecs

There are two major techniques for speech compression: Adaptive Differential Pulse Code Modulation (ADPCM) and Linear Predictive Coding (LPC). Different types of codecs are developed based on these techniques.

The ADPCM approach uses statistical properties of human speech to make a prediction about the size of the next sample based on previous information. Hence, a transmitter sends only the difference between a real value and a predicted value. The receiver uses the prediction algorithm and the difference to reconstruct the speech data (Harrington and Cassidy, 1999). ADPCM codecs are currently recommended for Voice over IP (VoIP) applications as specified by the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) in G.726 [18]. These codecs are sampled at 8kHz and have four different data rates: 40kbps, 32kbps, 24kbps, and 16kbps. These codecs attempt to achieve the uncompressed voice quality of PCM coding with lower bit rates.

LPC is introduced in the 1960s and is used for low bit-rate speech codecs. In LPC,

[15] The MPEG home page, www.chiariglione.org/mpeg.
[16] www.microsoft.com/windows/windowsmedia/
[17] www.real.com
[18] G.726, 40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM) , December 1990, http://www.itu.int/rec/T-REC-G.726/en

n^{th} sample is represented as a linear combination of a previous p plus a prediction error. LPC coding is more complex than ADPCM and results in higher processing delay (Harrington and Cassidy, 1999). Some common audio codecs based on the LPC technique are:

- RPE-LPC (Residual Pulse Excited LPC): this codec is used in GSM (Group Special Mobile) and has a data rate of 13kbps.
- CS-ACELP (Conjugate Structure - Algebraic Code Excited Linear Prediction): this codec is standardized in ITU-T G.729 [19] and has a data rate of 8kbps.
- MP-MLQ (Multi Pulse - Maximum Likelihood Quantization): This codec is standardized in ITU-T G.723.1 [20] and has a data of 6.3kbps.

These low bit-rate codecs are often recommended for voice transmission in mobile and wireless applications.

2.3.4.2 Transmission, IP Encapsulation, Codec-Related Processing Delay, and Bandwidth Overhead

Typically, each block of speech sample is processed into a compressed frame. This frame is then dropped into an IP packet. IP packets are then transported by using User Datagram Protocol (UDP) and Real-time Transport Protocol (RTP). The use of UDP and RTP for VoIP has been standardized in the H.323 recommendation for packet-based multimedia communications systems [21]. DICE (Boustead et al., 2005), the voice support system for the Mimaze (Bolot and Parisis, 1998), and common Instant Messagers, such as MSN and Yahoo Messenger also use this approach.

A key parameter that affects audio codecs-related processing delay and bandwidth transmission overhead is the number of codec frames encapsulated in an IP packet. As specified in ITU-T G.114 [22], for every coder, the delay due to audio codec-related processing in an IP-based system with modem or ADSL access bandwidth links is:

[19] G.729, March 1996, http://www.itu.int/rec/T-REC-G.729/en
[20] G.723.1, March 1996, http://www.itu.int/rec/T-REC-G.723.1/en
[21] http://www.openh323.org
[22] G.114, One-way transmission time, May 2003, http://www.itu.int/rec/T-REC-G.114/en

Coder type	Rate (kbps)	Frame size (ms)	Look-ahead (ms)	Delay introduced by coder-related processing (ms)	Reference
PCM	64	0.125	0	0.375	G.711
ADPCM	40	0.125	0	0.375	G.726
ADPCM	32	0.125	0	0.375	G.726
ADPCM	24	0.125	0	0.375	G.726
ADPCM	16	0.125	0	0.375	G.726
CS-ACELP	8	10	5	35	G.729
RPE-LPC	13	20	0	60	GSM 06.10 Full-rate
MP-MLQ	6.3	30	7.5	97.5	G.723.1

Table 2.1
Summary of common audio codecs and the codec-related processing delays in IP-applications (one frame per packet), specified in ITU-T G.114.

$$(2N+1) \text{ x frame size + look-ahead}$$

Where:

N: Number of frames in each IP packets.

look-ahead: the time a coder spends to look into the succeeding frame to improve compression efficiency.

Table 2.1 shows the delay incurred by codec-related processing when only one frame is stored in each IP packet. It is shown that ADPCM codecs have very small codec-related processing delay. All low bit-rate codecs have large codec-related processing delay (from 35 to 97.5ms). Also, these delays are only for the case that one frame is stored in each IP packet. When a number of frames are stored in each IP packet, these delay will increase proportionally.

The mount-to-ear delay for voice communication in a packet network is recommended in the ITU-T G.114. The recommendation specifies that the mount-to-ear delay of under 150ms is acceptable for most user applications. The delay in a range from 150ms to 400ms is acceptable provided that administrators are aware of the transmission time and the impact it has on the transmission quality of user applications. When the delay goes above 400ms, many or all users will be dissatisfied. In addition, some VoIP providers (Flak and Stumm, 2002) recommend that users do not

Codec	Codec data rate (kbps)	Nominal access bandwidth (kbps)
G.711	64	87.2
G.729	8	31.2
G.723.1	6.3	21.9
G.723.1	5.3	20.8
G.726	32	55.2
G.726	24	47.2
G.728	16	31.5

Table 2.2 Bandwidth requirements for several common VoIP audio codecs.

get distracted if the mount-to-ear delay is under 200ms.

In a VoIP application, the mount-to-ear delay consists of the network delays and the codec-related processing delay. The network delay consists of propagation delay and delays at routers and switches in the path from the voice transmitter to the receiver. Depending the type of codecs and the IP encapsulation scheme (e.g., frames per IP packet), the network delay tolerance of an audio codec can be determined. This delay is equal to the mount-to-ear delay tolerance (e.g, 150ms or 200ms) subtract the codec-related processing delay. With this regard, among these above speech codecs, ADPCM codecs has the smallest network delay tolerance. For example, assuming the mount-to-ear delay tolerance is 200ms, the network delay tolerance of the ADPCM codecs is 199.625ms while this delay for MP-MLQ codec is 102.5ms

Due to IP encapsulation, the access bandwidth requirement for transmitting an audio codec is significantly higher than the codec data rate. For example, when one audio codec frame is encapsulated in each IP packet, the bandwidth overhead for transmitting each frame include the link layer header, IP header, UDP header, and RTP header. Typical access bandwidth requirement of several common VoIP audio codecs are shown in Table 2.2[23]. As shown in the table, for low bit-rate codecs such as G.723.1 and G.729, the actual bandwidth requirement is nearly four times of the codec data rate. For higher bit-rate codecs such as G.726 and G.711, the bandwidth overhead is smaller.

[23] Voice over IP - per call bandwidth consumption, Cisco document ID: 7934, www.cisco.com

2.3.4.3 Choices of Audio Codecs

In the previous section, various audio codecs with delay tolerance and bandwidth requirements are presented. There is a trade-off between the codec data rate and the speech quality and codec-related processing delay.

Low bit-rate codecs result in low speech quality, background noise, and high codec-related processing delay. When being used in the IP network environment, the actual bandwidth requirement is considerably higher than the codec data rate. With regard to the immersive audio mixing application, low bit-rate audio codec is not capable of producing good quality spatial voice representation (Bolot and Parisis, 1998) due to the mixing operation.

DICE (Boustead et al., 2005) currently support several audio codecs. Experiments show that ADPCM codecs (G.726) are most suitable for the immersive voice application. In particular, the 32kbps ADPCM codec is commonly used. This audio codec achieves reasonable level of speech compression as well as good spatial voice quality delivered to users.

2.4 Infrastructure Support

This section reviews network and server infrastructures for the creation and distribution of real-time content. We discuss the trend of using shared server infrastructure for deploying various services and reducing the need for capital intensive infrastructure investment. Several research results related to shared server infrastructure such as provisioning and resource management are presented. In addition, we describe recent technologies and approaches that provide low latency and reliable network infrastructures to connect servers and deliver content to users, and improve the performance and scalability of applications using these servers.

2.4.1 Server Infrastructure

In this section, we describe several shared server infrastructures, with a particular focus on distributed servers that support real-time creation of content such as mul-

tiplayer online games. We also discuss issues related to provisioning and resource management.

2.4.1.1 Shared Server Infrastructure

In the current Internet, the use of server infrastructures has become popular. In general purpose computing, server platforms have advanced from a single central server to cluster and grid computing. Multiple applications can run on these server infrastructures. In web applications, techniques has been developed to improve the scalability (in the case of clustered web servers), or both scalability and network latency (in the case of CDNs) of content delivery. In multiplayer online games, several server hosting infrastructures such as Butterfly Grid, have been designed to improve the scalability and reliability of the game deployment.

In addition, these server infrastructures are often *shared* between different applications or parties (e.g., service providers). The key benefit of shared infrastructure for service providers is to reduce the need for capital intensive infrastructure investment. In addition, these *virtual resources* such as virtual server capacities can be suitably adjusted based on the business requirements. This factor is very important for MMOG, which require considerable infrastructure investment. Due to the difficulty in predicting the success of a new MMOG, it is risky for game publishers to deploy dedicated server and network resources. Several shared server infrastructures that enable real-time content creation are described as follows.

An architecture called Switched Overlay Network (Boustead et al., 2004) has been designed by researchers in our group to provide infrastructure support for the provision of scalable delay-constrained distributed applications. Example of these applications include multiplayer online games and real-time composition of multimedia from geographically distributed sources in the Internet. SWON supports these applications by the use of a multitude of servers and the underlying network connecting these servers. In this model, an application provider (e.g., a game service provider) hires *virtual servers* from a number of *server providers* over a large geographical span. Interconnection of virtual servers is done by using the underlying network infrastructure provisioned by various network providers. By using geographically

distributed servers and efficient routing path between servers, SWON aims to improve latency in communication delay between clients in interactive applications such as multiplayer online games. The network infrastructure for connecting servers in SWON will be discussed in more detail in Section 2.4.2.3.

The research efforts in (Whitaker et al., 2002) (Fraser et al., 2001) have developed operating system techniques to allow different clients to share the same network based servers securely and efficiently. This server infrastructure can be geographically distributed so that it enables clients to choose a suitable processing location for minimizing communication latency. The research project in (Peterson et al., 2002) investigates the use of this shared server infrastructure to form a service oriented network. This research shows how distributed virtual machines can collaborate together to form an overlay network and deliver a specific service.

The research project in (Shaikh et al., 2004) presents a prototype implementation of a shared and on demand service platform for multiplayer online games. The system monitors game and system performance and automatically provision server resources in response to changes in the game workload conditions. Several performance metrics such as CPU utilization (including instantaneous and smoothed metrics) and server processing latency are evaluated for resource allocation decision. The experimental results show that the session arrival rate is a key factor in determining the suitability of different metrics in provisioning. In particular, smoothed metrics can avoid unnecessary overprovisioning in short-term workload changes but not suitable when the session arrival rate is high. This research project demonstrates the feasibility of applying utility computing concept to an infrastructure for hosting multiple multiplayer online games as well as other business applications.

Grid Computing

A notable example of a shared server infrastructure is grid computing. Grid computing technologies were originally developed to enable resource sharing within scientific community for computationally demanding data analyses. Recently, grid computing has shown some benefits for commercial distributed applications, such as infrastructure for massive multiplayer games, and other e-business computing appli-

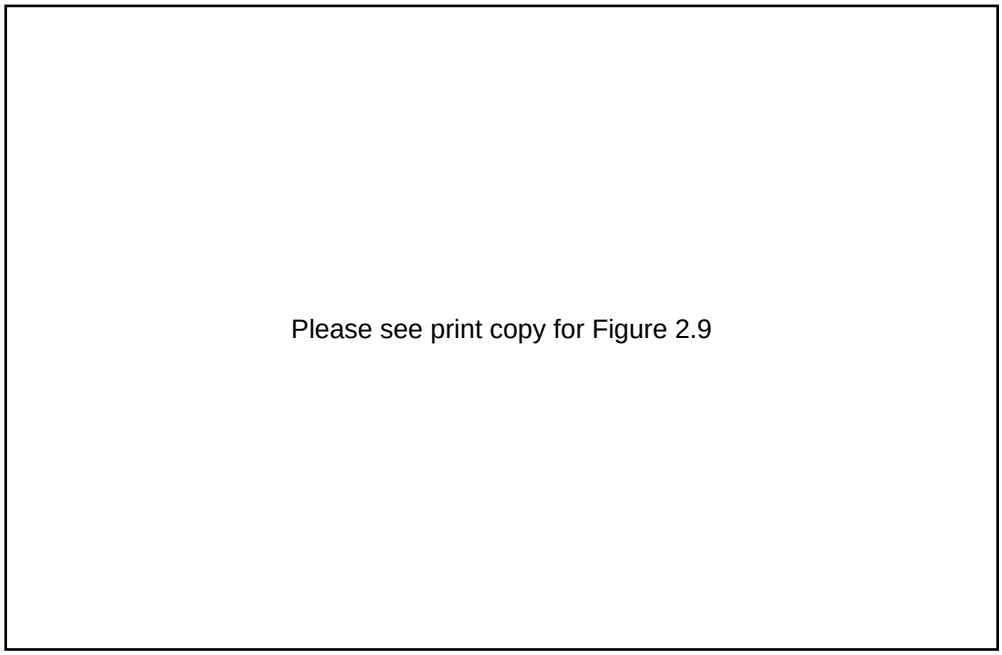


Figure 2.9 The layered Grid architecture (Foster and Kesselman, 2004).

cations over the Internet (Carpenter, 2003). In a grid, each participant is referred to as a *Virtual Organization* (VO). Grid technologies provide mechanisms for sharing and coordinating the use of diverse resources in VOs, and thus enable the creation of *virtual* computing systems that are integrated to deliver the desired QoS (Foster et al., 2001).

The general Grid architecture (Foster and Kesselman, 2004) is based on the principles of the "hourglass model" as shown in Fig. 2.9. The functions of these layers are summarized as follows.

- *Fabric:* At the bottom, the *fabric* layer provides the resources to which shared access is mediated by Grid protocols, for example, computational resources, storage systems, network resources, and sensors.
- *Resource and connectivity protocols:* The narrow neck of the hourglass defines a small set of resources and connectivity protocols, which facilitate the sharing of individual resources. The number of protocols in this layer is small compared to the fabric layer and the collective services and user applications layers. Communication protocols in this layer enable exchange of data be-

tween fabric layer resources. In current Grid architectures, these protocols are mainly drawn from the TCP/IP protocol stacks: specifically, the Internet (IP and ICMP), transport (TCP, UDP), and applications (DNS, OSPF, RSVP).

- *Collective services:* This layer contains protocols and services not associated with any one specific resource but instead capturing interactions across collections of resources. Some of the major functions of this layer are: *Directory services* to allow VO participants discover the existence and properties of VO resources; *Coallocation, scheduling, and brokering services* to allow VO participants to reqest the allocation of resources; *Monitoring and diagnosis services* to support the monitoring of VO resources.

- *User applications* This layer comprises user applications that operate within a VO environment.

An example of a grid computing system that is specifically designed for massive multiplayer games is the Butterfly Grid [24]. The grid has two main locations in East and West of USA, each location has a large number of servers. Multiple games can be hosted by the grid. The grid provides on-demand computing and allows game service providers to avoid huge up-front server infrastructure investment. In each game, the virtual world can be partitioned into smaller areas, each is handled by a server in the grid. As claimed by the Butterfly Grid, this approach can enable the game to scale up to a million participants. The server structure itself is also scalable since new servers can be inexpensively added to the grid.

2.4.1.2 Server Provisioning

Provisioning is a key issue in shared server infrastructure for supporting multiple applications and customers. The purpose of provisioning is to choose a suitable server or a set of servers for each application or customer. There are various performance metrics to be considered for provisioning. These include server load, server cost, network cost and network latency. When servers are located at the same location (e.g., a cluster of server), the provisioning decision is mainly based on load bal-

[24] Butterfly Grid, http://www.emergentgametech.com

ancing and load sharing purposes. On the other hand, when servers are located in different parts of the Internet, apart from server resources, the latency and network cost between these servers and between servers and clients are also key provisioning factors. In this section, we outline some research in provisioning geographically distributed servers.

The research in (Safaei et al., 2001) presents a resource provisioning model for a shared server infrastructure that supports a large number of service providers. This model enables the provision of both network bandwidth and computation resources efficiently. In particular, the optimization algorithm in this model configures each service (e.g., video transcoding) with a suitable server or a suitable set of servers based on the network cost between the clients and the servers and the processing cost for initiating the service at each server. A related research in (Choi et al., 2003) presents a general approach to the problem of configuring path for content transformation applications (e.g., transcoding and compression/decompression) which requires intermediate processing in the network. The path configuration is based on shortest path routing algorithms which consider both processing cost at a server and network link cost in both capacity-constrained and unconstrained approaches. This research study enables efficient use of network and computation resources by configuring application sessions with suitable servers and routing paths.

Another study in (Choi and Shavitt, 2003) discusses general problem of placing servers for different services in the network. For each service session, this study considers the network costs that connecting participants to a server and a cost of initiating the service at each server. The authors formalize the problem of optimally placing servers and introduce approximation algorithms. A new angle of this study is to look at two interesting problems: finding a limited number of servers to obtain the maximum gain, and finding the minimum number of servers needed to achieve a given level of service. As it will be shown later in the thesis, the number of servers also has a significant effect on the latency of distributed game service architectures.

In CDNs, there is a need to choose optimal locations for replicating web objects. There are considerable research results on replica placement algorithms for CDNs, aiming to optimize latencies and costs based on client information and potential

server distribution. The research in (Qiu et al., 2001) proposes several placement algorithms that use workload information, such as client latency and request rates, to make placement decisions. Another study in (Cronin et al., 2002b) proposes placement algorithms subject to the constraint that replicas can be placed only at certain locations. The work in (Nguyen et al., 2003b) proposes an optimization model to provision multiple CDNs in a shared infrastructure.

The provisioning of distributed game servers have been discussed in (Lui and Chan, 2002) (Ta and Zhou, 2003) and references therein. These research efforts provide mechanisms to partition the game virtual world to smaller areas and assign each area to a suitable servers. However, these works are mainly concerned with load balancing and load sharing purposes. The provisioning of geographically distributed servers for improving latency for the game is not addressed.

In short, most of provisioning approaches for shared server infrastructure are based on a cost model with an emphasis on optimizing resource usage. There are some research that considers delay such as client latency when retrieving content in a CDN. However, to the best of our knowledge, the provision of delay-constrained interactive applications such as multiplayer online games and communication information processing for these games is not adequately addressed. Especially, in MMOG, there are various provisioning issues need to be considered including resource scalability and latency in group communication. There is a need for specific studies on the provision of multiplayer online games and voice communication support for these games.

2.4.1.3 Server Resource Management

In a shared server, sharing of server resources between different applications and users is a key issue. There have been various techniques and approaches for the management of server resources. Depending on the type of applications running on a shared server, each system has a different way of managing resources.

There are several commercial server systems that support virtual hosting. These include the Solaris resource manager [25], Ensim ServerXchange [26], and Apache virtual

[25] Solaris resource manager, http://www.sun.com/blueprints
[26] Ensim ServerXchange, http://www.ensim.com

hosts [27]. These systems allow several web servers to be hosted in the same physical server and provide low-level support for partitioning resources. Each web server is allocated a portion of server resource which may include CPU time, memory, and network bandwidth.

In addition, research projects in (Banga and Druschel, 1999) (Reumann et al., 2000) and references therein have proposed several server resource management schemes. These schemes aim to provide more fine-grained and application-specific control of server resources compared with general-purpose server operating system. They can fall to two categories: *resource-oriented* and *service-oriented*. The research in (Banga and Druschel, 1999) creates a new operating system abstraction called a *resource container* which encompasses basic CPU and network shares allocated to each customer (e.g., a company web site). This is referred to as resource-oriented since the resource binding and scheduling threads is based on multiplexing among different resource containers. On the other hand, researchers in (Reumann et al., 2000) proposes a service-oriented approach where resources are managed based on types of services. In this approach, per-service resource partitioning is used and each service may be shared among different customers.

In server resource management, scheduling of processing resource is a key task. Traditional operating system generally use time sharing scheduling algorithms, which allocate CPU resources among processes based on average time sharing. For the processing of real-time content transformation such as video compression/decompression and media transcoding, time sharing algorithms may not be efficient. Since switching between different applications based on average CPU time, the processing of a packet may stop and resume after a period of time. This may cause large processing overhead as well as a substantial server processing delay which is not acceptable for real-time applications.

As a result, researchers have proposed several packet-based processing resource scheduling algorithms for real-time applications. In this case, a packet or a video frame is the smallest processing unit for scheduling. Current packet scheduling algorithms used in network bandwidth provisioning often rely on obtaining packet length

[27] Apache virtual host documentation, http://httpd.apache.org/docs/vhosts/

from the packet header to determine the scheduling decision. However, it is generally difficult to determine packet processing time in advance for processing resource scheduling. Several approaches has been proposed to address this problem. Research efforts in (Pappu and Wolf, 2002) (Sabrina and Jha, 2003) and references therein use estimation of a packet processing time for scheduling. This may be suitable for a limited number of applications. For other applications, it is possible to design a packet scheduling algorithm without using prior knowledge of processing time as demonstrated in (Goyal et al., 1996). A trade-off of this approach is a higher delay penalty. In short, this is an interesting area of research and need to be addressed further. Especially, there is a need to investigate the server sharing policy for emerging applications that require real-time content creation.

2.4.2 Network Infrastructure Support

Real-time contents including game state information and communication information (e.g., audio streams) often require certain QoS metrics such as low latency and low loss rate. As discussed earlier, to improve the scalability and latency of real-time content creation and delivery, there is a trend of moving from the central server model to a distributed server model. In a distributed server infrastructure, a multitude of servers are located in different locations in the Internet and may be shared among different applications.

In this section, we describe technologies and approaches for connecting these servers to deliver services. We focus on overlay networking approaches which are based on implementation of functionality at end systems and requires no change in the underlying network. These approaches provide features that are not available in current IP network such as application-level routing and multicast.

2.4.2.1 Overlay Networks

A simple but expensive approach for providing QoS connectivity between servers is to use a virtual private network. A Virtual Private Network (VPN) connects geographically dispersed sites with security and/or QoS capabilities (Gleeson et al., 2000). However, deploying a VPN requires a network provider to configure routers/switches

in order to set up topology and QoS links. In addition, it is also difficult and expensive to deploy VPN infrastructure across many different network domains.

An overlay network is defined as a network of nodes and virtual links that operates on top of an existing network in order to provide a new service that is not available from that existing network (Stoica, 2004). The advantage of an overlay network is that it provides a new service *without* changing software/protocols in the underlying networks. This general definition of overlay networks is very broad. For example, an VPN can be considered as an overlay network of IP network to provide security and/or QoS capabilities. Hence, this section will concentrate only on recent overlay network approaches to implement advanced network services that are not available in the current IP network. Specifically, recently, academics and industry have put tremendous efforts into the use of overlays as a fundamental approach for developing new network functionality (e.g., multicast) and improving performance and availability of existing applications (e.g., by using application-layer routing).

An example of an overlay network that focuses on fault detection and fast rerouting is Resilient Overlay Network (RON) (Andersen et al., 2001). RON consists of a multitude of nodes located in different locations in the Internet. These nodes monitor the functioning and quality of Internet paths among themselves, and use this information to decide whether to route packets directly over the Internet path or over other RON nodes, optimizing application-specific routing metrics. Measurements in this work shows that RON was able to detect, recover and route flows from path outages in less than twenty seconds on average while today's wide-area routing protocols such as BGP take at least several minutes. Furthermore, RON was able to improve the loss rate, latency or throughput perceived by data transfers. As indicated in (Mauve et al., 2002), a game proxy system can utilize RON approach for fault detection and rerouting between proxies for improving game performance.

An overlay architecture called OverQoS (Subramanian et al., 2004) aims to improve the loss rate and network bandwidth in multimedia delivery and multiplayer online games. This architecture uses an abstraction called *Control Loss Virtual Link* (CLVL), which provides statistical loss guarantees to a traffic aggregate between two overlay nodes at varying network conditions. CLVL detect packet losses and retrans-

mit them between overlay nodes, improve packet losses at application levels. Also, it enables overlay nodes to control the bandwidth and loss allocations among the individual flows within a CLVL. The advantage of this architecture is demonstrated in two applications. First, RealServer [28] can use OverQoS to improve the quality of multimedia delivery by protecting more important frames at the expense of less important ones. Second, a multiplayer game such as Counterstrike can use this architecture to avoid frame drops and prevent end-hosts from getting disconnected in the presence of high loss rates. The trade-off is the redundancy bandwidth overhead.

Multicast is important for group communication in DVE. Although IP multicast was introduced in the late 1980s, it has not been widely deployed in the Internet due to its scalability, network management, deployment and support for higher layer functionality (Chu et al., 2000). Mbone was one of the earliest overlay networks on the Internet that uses a tunnelling technique to do multicast between geographically disjoint areas where IP multicast was possible. MBone is an outgrowth of the first IETF "audiocast" experiment in 1992 (Casner and Deering, 1992), in which live audio was multicast from the IETF meeting site to destinations around the world. IP multicast packets are encapsulated for transmission through tunnels so they appear as unicast packets to intervening routers that do not support IP multicast. When these packets arrive at the other end, encapsulating IP headers are removed and these packets are forwarded based on the IP multicast headers.

Recently, several projects in (Chu et al., 2001) (Banerjee et al., 2002) and references therein have developed a number overlay multicast architectures as an alternative to IP multicast. As shown in Fig. 2.10, in overlay multicast (also known as application layer multicast), multicast functionality is implemented at end systems while IP multicast functionality is implemented at IP routers. Each of the overlay nodes acts as an intermediate IP router to efficiently distribute content along a pre-defined mesh or tree. Overlay multicast is generally proposed for media streaming and content delivery. In multiplayer online games, overlay multicast can be used for distributing state information and communication information between servers in a distributed game server architecture or between proxies in a game proxy system.

[28] Helix Universal Server, www.real.com

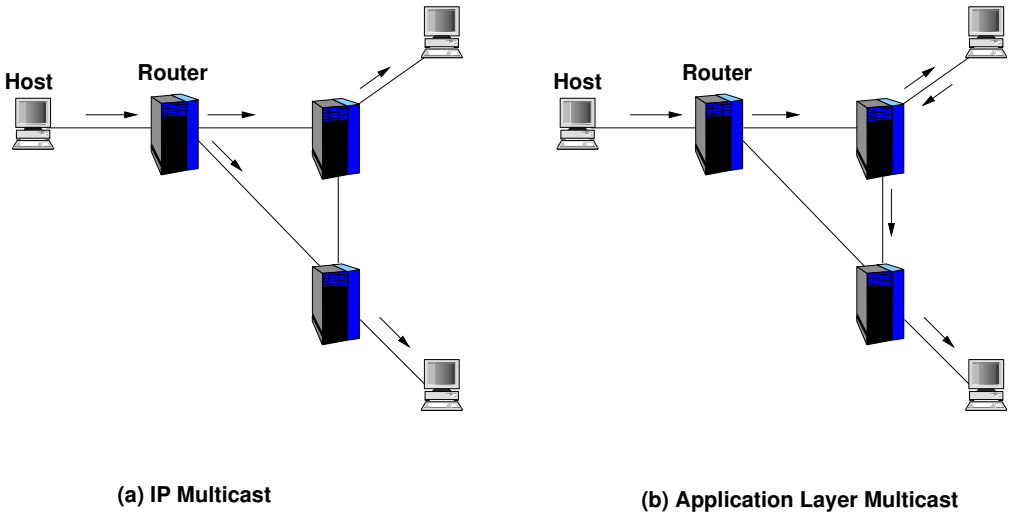


Figure 2.10 Example of IP multicast and application layer multicast.

The previous overlay architectures are built over the best-effort Internet without any QoS support from the underlying infrastructure. The service overlay network (SON) (Duan et al., 2003) framework provides end-to-end value-added Internet services by purchasing bandwidth with certain QoS guarantees from individual network domains via bilateral service level agreement to build a logical end-to-end service delivery infrastructure on top of the existing data transport network. As shown in Fig. 2.11, a SON can request bandwidth guarantees between any two service gateways across a network domain. Relying on the bilateral SLAs, a SON can deliver end-to-end QoS service to its users via appropriate provisioning and service-specific resource management. In this way, the SON architecture decouples application services from network services, hence, reducing the complexity of network service management and control. The network domains are simply concerned with provisioning of data transport service according to SLAs. On the other hand, due to its service awareness, a SON can deploy service specific provisioning, resource management and QoS control mechanisms to optimize its operation for its service. SON architecture is particularly applicable to the distribution of delay sensitive real-time content across geographically dispersed network domains in the Internet.

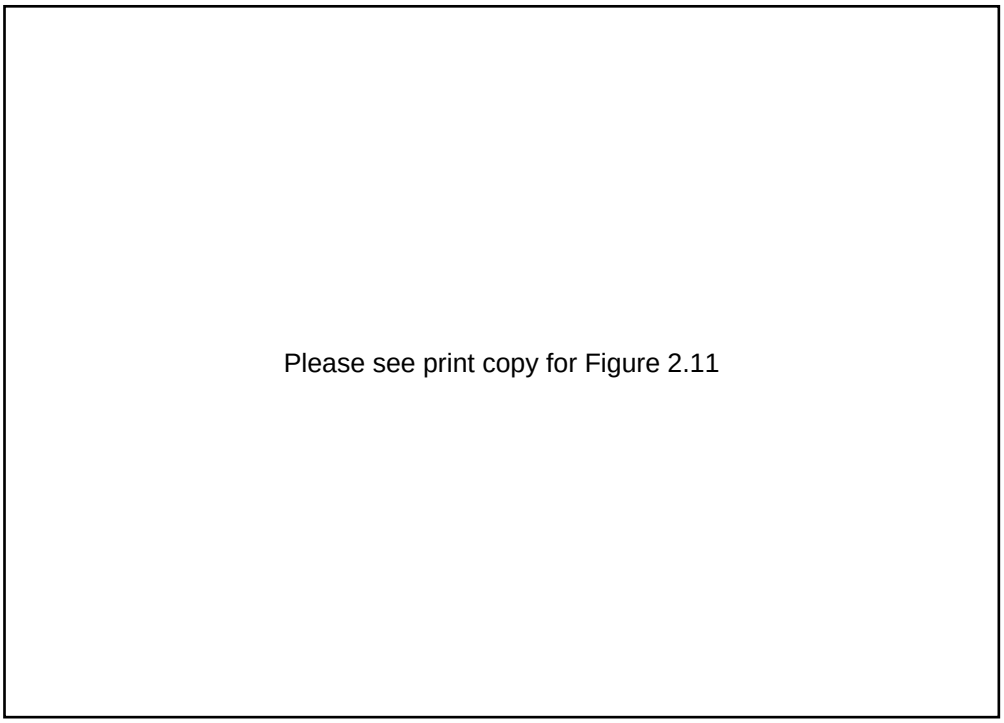


Figure 2.11 Illustration of a service overlay network, adapted from (Duan et al., 2003).

2.4.2.2 Middle-aware Support Overlay

Various functions are required at each overlay node. These may include packet classifier, application level routing and filtering. The implementation of these functions in the architectures discussed earlier such as OverQoS (Subramanian et al., 2004) and RON(Andersen et al., 2001) are based on software at end systems. These implementations are suitable for a small-scale overlay network. Other implementations are possible.

The community has settled on the name of middleboxes (Carpenter and Brim, 2002) to refer to a device that implement the above functionalities. Middlebox is loosely defined as any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host. A range of middlebox functions include: address translation, packet classifier, proxies, firewall, and application level routing. These boxes are typically located at network edges and implemented in an ASIC or in software. While an ASIC re-

duces the functional flexibility, software implementation has limited throughput and may not be suitable for large-scale applications that require high throughput. These applications may include caching, filtering and forwarding game events in massive multiplayer online games, and intelligent forwarding for large-scale peer-to-peer systems

Recent research projects advocate the use of network processor to build middleboxes which is a mid-point between an ASIC (a pure hardware solution) and pure software solution. The research efforts in (Bauer et al., 2002) (Rooney et al., 2003) develop a programmable network middleware support called a "booster box" for building overlay networks to support large distributed applications, including scalable servers for massively multiplayer games and large peer-to-peer systems. As indicated in Fig. 2.12a, a booster box is a general purpose computation platform with an interface that allows application specific logic to configure the underlying packet forwarding engine. It consists of two abstraction layers: a booster layer and a data-forwarding layer. The booster layer consists of a set of boosters, each booster is a combination of application-specific logic and generic booster library functions. The library contains the API through which boosters can call the data layer operations. The booster control point coordinates the boosters and manages any data that is common to all of them. The data forwarding layer does simple forwarding as well as copying or diverting selected traffic to the booster layer and is implemented by using network processors.

As an example, a booster can contain game application specific codes for intelligent forwarding of game events. When a packet containing game events arrive at the booster box, the packet classifier at the data forwarding layer determines whether this packet belongs to a particular booster. After the packet is identified, the game specific codes located in the booster will trigger the operation on this packet at the data forwarding layer, such as whether to copy or forward the packet to a suitable destination. This operation is done by the API contained in the booster library. By implementing this intelligent forwarding at various booster boxes located at network edges, game state events are handled locally, hence, the scalability and response time are improved. Therefore, in this case, booster box is said to "boost" the game

performance compared to current approaches of using game servers alone.

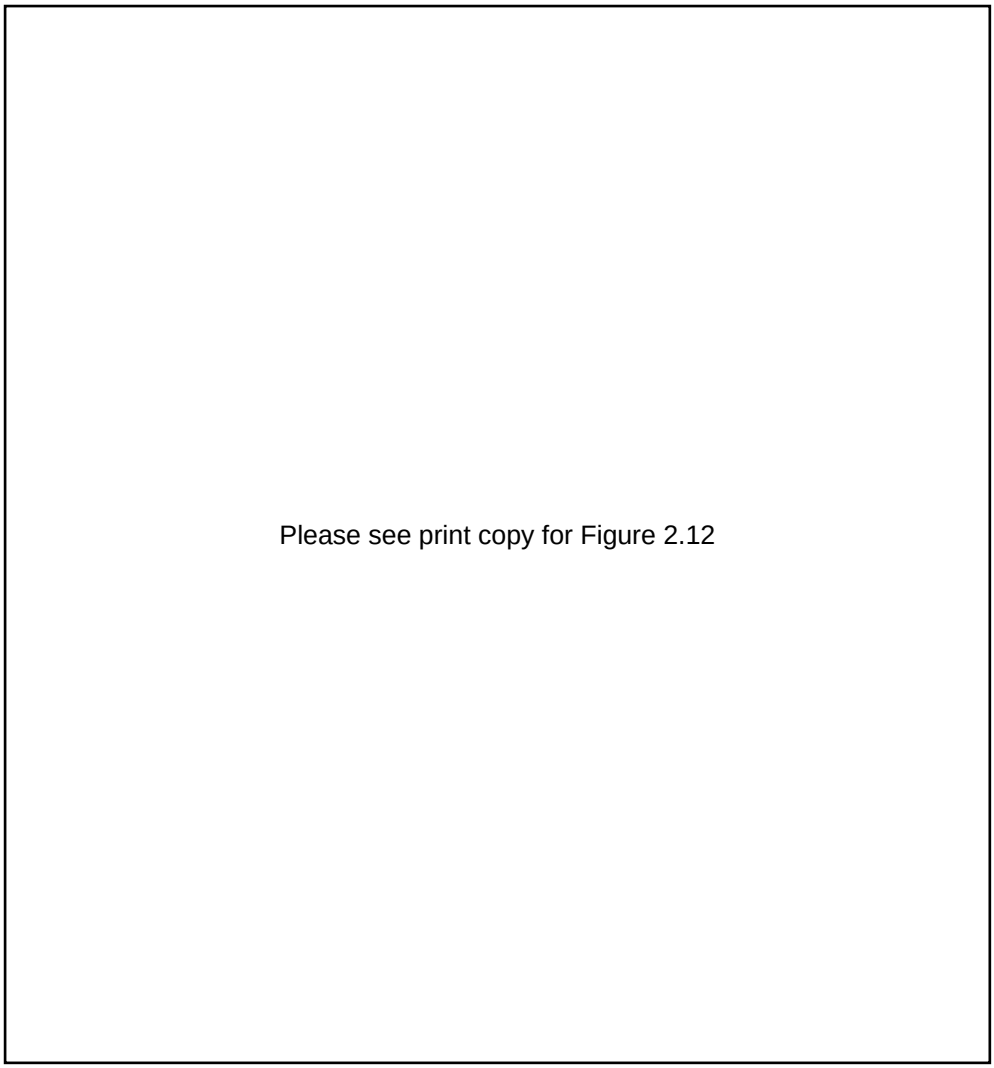
Different booster boxes located in various locations in the Internet form a booster overlay network (BON) (Bauer et al., 2002). This overlay network can provide routing functions for low-delay and high throughput traffic. A number of applications can be hosted in BON including multiplayer online games. These applications share the BON base topology but can have different endpoints in BON. BON is particularly suitable for enhancing the scalability and reliability of massively multiplayer online games. As shown in Fig. 2.12b, booster boxes are used for supporting a distributed game server architecture. Each booster serves a number of game clients in its network vicinity and perform caching, filtering, forwarding, and redirecting game events in a game specific way.

2.4.2.3 Switched Overlay Network

In section 2.4.1.1, the Switched Overlay Network (SWON) (Boustead et al., 2004) has been introduced as an architecture that enable the use of shared server infrastructure. This section describes the network support for connecting servers and enabling application level routing in SWON.

Fig. 2.13 shows an example where an application provider purchases differen types of QoS guarantees in each network domains depending on the service availability, cost and application requirements. There are two mechanisms for interconnection of servers: *Internet connectivity* and *overlay connectivity*. Internet connectivity is the direct Internet routed path between servers. Overlay connectivity is established by the creation of tunnels between server sites. SWON allows applications to choose between these two connections according to their requirements such as latency.

The use of QoS overlay paths over Internet paths is supported by the fact that many default paths in the Internet have potential alternate paths with much lower latencies. This is investigated in the work in (Savage et al., 1999) which shows a measurement-based study comparing the performance seen using the “default” path taken in the Internet with the potential performance available using some alternate paths. The results show that, for 30-80% of the cases, there is an alternate path with superior qual-



(a) Distributed game servers with booster boxes

Figure 2.12 Booster box and the deployment of booster boxes for distributed game servers, adapted from (Rooney et al., 2003).

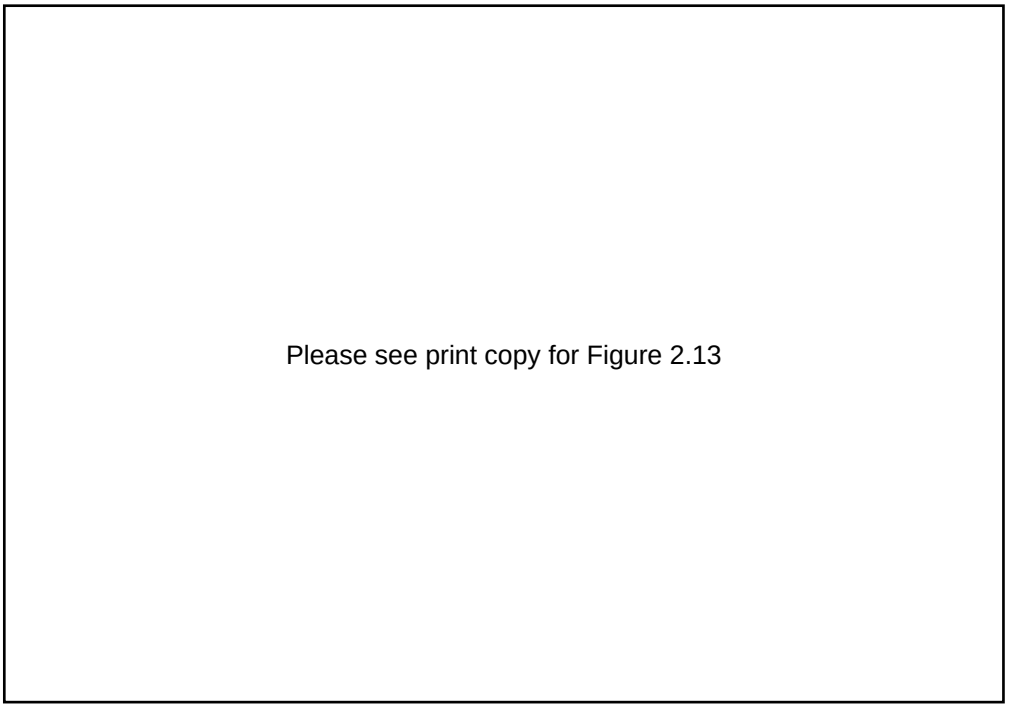


Figure 2.13 Overlay server and routing between these servers over multiple network domains (Boustead et al., 2004).

ity. In particular, these alternate paths can provide lower round-trip latency and/or lower loss rate. This is due to the fact that current wide-area routing protocols, such as BGP, are mainly concerned with the exchange of connectivity information and do not incorporate measures of round-trip time or loss rate into account. In addition, economic considerations can also limit routing options as some parts of the Internet refuse to carry traffic without a contractual agreement (Savage et al., 1999).

To enable application controlled Quality of Service, packet forwarding capabilities are required at server sites. This allows an application provider to control packet forwarding of their flows depending on the application requirements, e.g., by using an overlay path or on Internet path. In addition to controlling the QoS routing of flows, other packet operations can be considered at server sites including multicasting, request routing and fast session redirection.

In order to provide switching over a shared server infrastructure, a hardware based implementation called *tunnel switch* is proposed. Compared to software based imple-

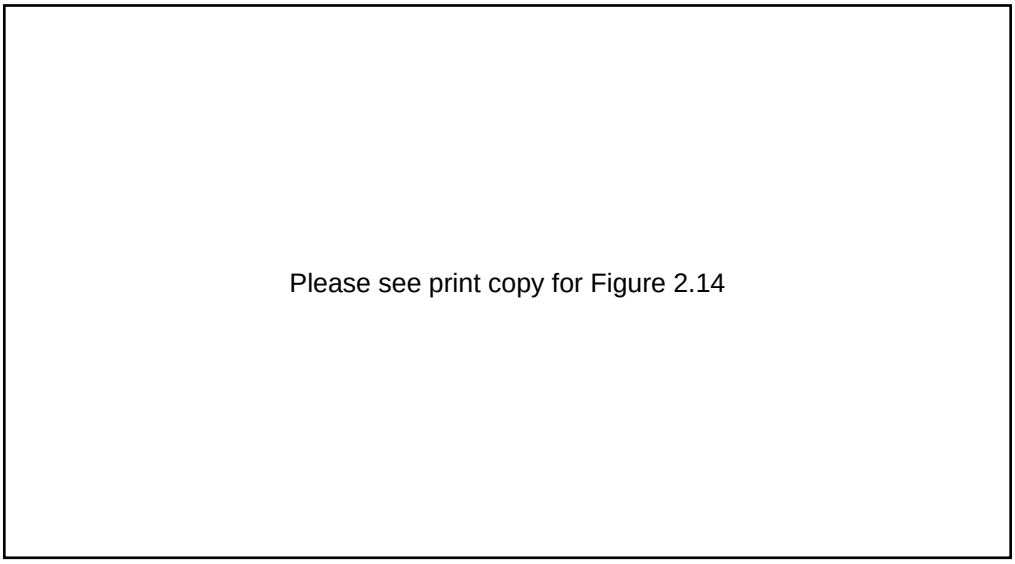


Figure 2.14 Tunnel Switch design (Boustead et al., 2004).

mentation, this hardware based implementation can provide lower packet forwarding delay and better scalability. By using the tunnel switch, SWON provides a secure virtual switch for each application provider, allows them to have application level control of hardware packet forwarding, duplication, filtering and so forth. As shown in Fig. 2.14, the switch has two ports, one connected to a server farm and the other one connected to the network provider. Each application provider owns a set of tunnels and one or more virtual servers within the server farm. Examples of operations of the tunnel switch for each application provider are shown in Fig. 2.14. These include: 1 - Switching between different tunnels; 2 - Switching between a tunnel and a virtual server; 3 - Switching between a tunnel and an Internet routed path; 4 - Switching between a virtual server and an Internet routed path.

SWON uses two levels of forwarding granularity by using two types of labels: forwarding labels and application labels. A forwarding label is set when a packet comes to the edge of SWON based on the destination address. The application provider may purchase a set of forwarding labels at each tunnel switch and use these to create an overlay. The application label is based on application information in layer 3 to layer 7 (e.g., the characteristic of a flow: real-time or non-realtime). Switching based on application labels increases the granularity of forwarding, allowing each application provider to have different QoS treatments for their own flows. For example, real-time

flows are switched to low latency uncongested paths while best effort flows may go to a longer path with possible presence of congestion. SWON allows each application provider to have their own label space and provides an API for each application provider to change the label table. As a result, each application has their own customized routing and can dynamically change the forwarding of their flows based on application requirements.

2.5 Conclusions

This chapter reviews the evolution of web content distribution, future development of content creation and delivery, and the infrastructure support to enable this development. We first describe the evolution of web content distribution that includes proxy caching, web clusters, content distribution networks and advanced content services. We then discuss the future development of content creation and delivery which requires real-time creation and delivery of content from a dynamic set of dispersed sources. This includes state information and communication information processing in distributed virtual environments. Finally, we describe server and network infrastructure for supporting this future development. In particular, we discuss the use of shared server infrastructures and describe approaches in provisioning and resource management of shared servers for supporting content creation and distribution. In addition, several overlay networking approaches are introduced to provide network support to connect servers and deliver content.

2.5.1 Issues Considered in Thesis

Deficiencies in existing literature and the issues considered in this thesis to address these deficiencies are presented as follows.

- Evaluate different delivery architectures for the provision of immersive voice communications to multiplayer online games based on existing game server architectures.
- A game simulation model for MMOG which consists of player distribution

in the Internet and different avatar distributions in the virtual world does not appear to be available in the literature. This thesis will provide this model.

- The provision of an immersive voice communication service to MMOG has not been adequately examined in the literature. A key issue in provisioning is how to assign players to servers. This thesis will devise server assignment algorithms for each of the proposed delivery architectures.

- There are several server assignment algorithms for distributed game server architectures. However, these algorithms are designed for load balancing and load sharing purposes. This thesis aims to design new server assignment algorithms with the emphasis on minimizing latency of group communication in the game.

- A quantitative investigation of different delivery architectures for the provision of an immersive voice communication service has not been carried out in the literature. This thesis will evaluate the performance of these delivery architectures.

- Several research projects have examined packet-based processing resource scheduling algorithms for content transformation applications such as multimedia transcoding and encryption. With the inclusion of the immersive audio mixing application, it is important to investigate the sharing of this application and other content creation applications. This thesis will design a new processing resource scheduling algorithm and examine the performance of this algorithm in providing resource sharing for various applications that require real-time content creation.

Chapter 3

An Immersive Audio Communication Service for Multi-Player Online Games

3.1 Introduction

As discussed in Chapter 2, there have been various research efforts in providing immersive voice communications to distributed virtual environments. However, a complete classification of delivery architectures as well as quantitative performance evaluation of these architectures do not appear to be available in the literature. Hence, this chapter introduces different delivery architectures for the provision of an immersive voice communication service to multiplayer online games. We outline main functions, advantages, and limitations of each architecture. This chapter also presents a game simulation model for evaluating the performance of these architectures.

This chapter is organized as follows. Section 3.2 introduces the concept of an immersive voice communication service for multiplayer online games. In Section 3.3, we introduce several delivery architectures for this service and describe limitations and advantages of each architecture. In Section 3.4, a simulation model framework is proposed to capture game delivery scenarios including player distributions in the physical network and avatar distributions in the virtual world. This section also describes key parameters and assumptions used for the performance study of delivery

architectures. Finally, conclusions are drawn in Section 3.5.

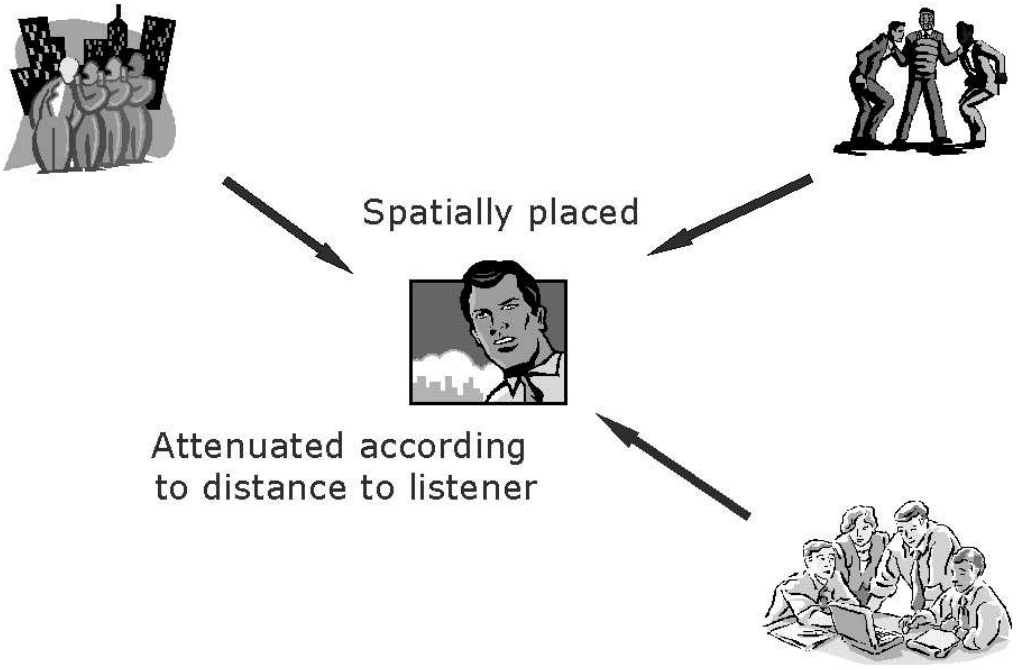
3.2 Concept of Immersive Voice Communication Service

We believe that certain classes of multiplayer online games will be far more attractive if *immersive* voice communications can be provided to game players. Using this service, an avatar can hear voices of the avatars it talks to as well as the background sound of other avatars in its area of interest or hearing range. Creating a personalised mix of voices of other avatars produces the audio scene, spatially placed and attenuated according to the distance to the listener. This means that a player can hear the sound coming from the direction of the speaker. The volume of the sound is adjusted based on the virtual world distance between the speaker and the listener.

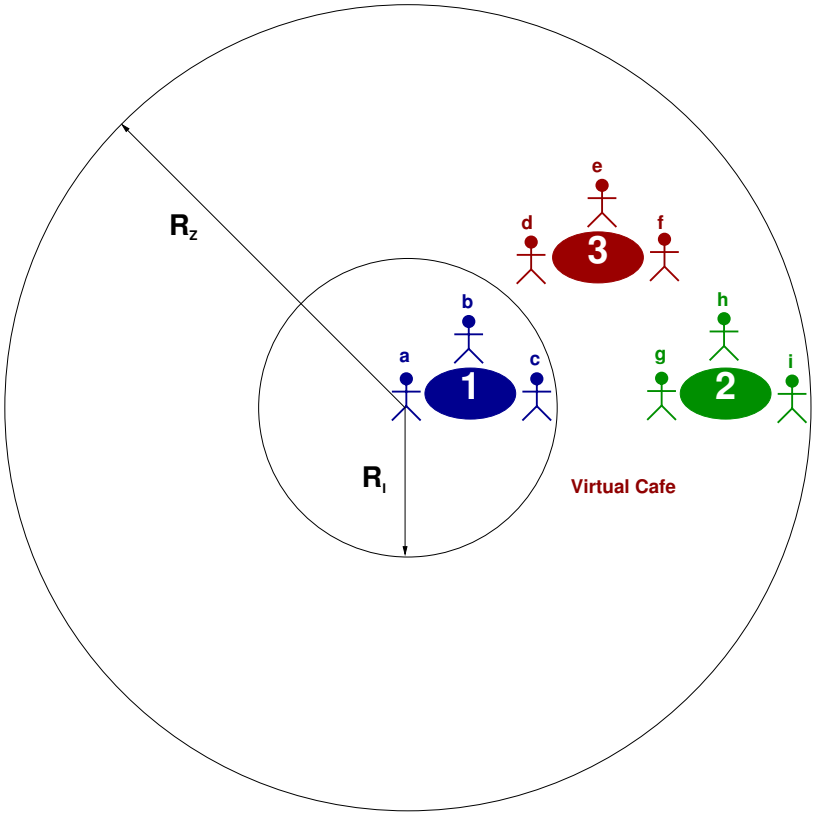
Although accuracy is desired, there is variation among the avatars about the importance of different audio signals. Let us define the *interactive zone* as the immediate vicinity of the avatar where active communicative interaction may take place, while the *background zone* as the region outside the interactive zone stretched to the limits of hearing range. In Fig. 3.1, the interactive zone is denoted as a small circle around each avatar and the background zone is denoted as a larger circle.

In reality, a player can hear a loud noise from large distance or the sound propagation can be blocked by virtual environment subjects such as walls and buildings. Hence, the interactive and background zone are not constant but depend on the magnitude of the sound sources and the structure of the virtual environment. This presents a number of challenges in developing an accurate simulation model of the virtual world. Therefore, we assume the constant sound excitation volume for the ease of modelling and we make this assumption as an initial attempt of a difficult piece of work.

Naturally, any audio information outside the hearing range can safely be ignored. But the important question is whether a realistic creation of the background zone is needed. In some cases, the user may only be interested in the interactive zone. The



(a) Immersive voice in a crowd scenario



(b) Zone definition

Figure 3.1 Immersive voice communication scenario and zone definition

background audio scene, therefore, appears as 'noise' and can either be blocked or simulated by a background chatter of suitable volume. However, there are situations where the multi-person voice communication is either the primary purpose of gathering in the virtual space or at least a very important means for achieving the actual goal. This will be particulary important for the genres of social games and interactive entertainment. Natural multi-person communication is often characterised by the presence of multiple simultaneous conversations among the people gathered in an environment such as a cocktail party, cafe, foyer of a conference, or market place. The ability to pick up interesting conversations in one's vicinity and join these groups or simply be aware of the peripheral discussions is critical to our sense of satisfaction of being in the presence of a crowd. The current style of audio teleconferencing, when a strict protocol of 'one at a time' for conversation has to be followed, may be too restrictive for crowded virtual spaces. We, therefore, believe that at least in certain situations, realistic presentation of the background audio zone is desirable, if not critical and this is a key differentiator of the immersive audio communication service compared to the existing solutions. Nevertheless, even in these situations, it is reasonable to assume that the accuracy requirements (in terms of delay and spatial placement) are more stringent for the interactive zone compared to the background zone.

Scalable provision of the immersive voice communication service for large virtual environments and over a large-scale infrastructure is extremely demanding on network and processing resources. The audio mix must be suitably composed from the relevant sources (avatars in one's hearing range) and personalised based on the perspective of each individual. A separate mixing computation is therefore needed for each participant. All this should be done in real-time and delivered with tolerable delay and reasonable quality. The application is also highly dynamic as people move (in some cases very frequently) in the virtual environment. This implies changes in the sources of audio signal that needs to be mixed for any given avatar.

We anticipate that the early commercial deployment of the immersive voice communication service will be for enhancing the networked multi-player games. In the future, the underlying technology can be used for creation of many applications in-

cluding collaborative working environments, teleconferencing services, and the like. A useful perspective is to view this as the next generation of voice communication, enabling multi-person audio communication capability (as opposed to telephony that is primarily person-to-person).

3.3 Basic Delivery Architectures

This section introduces basic architectures for delivering the immersive voice communication service to multiplayer online games. While these architectures can be applied to multiplayer online games in general, we focus on evaluating these architectures for massively multiplayer online games, which have a large geographic distribution of participants and are more challenging in terms of design requirements. The following architectures are considered for provisioning the immersive voice communication service: peer-to-peer, central server, distributed locale servers, and distributed proxies.

3.3.1 Peer-to-peer

In the peer-to-peer delivery architecture, all audio scenes are created on participants' devices. Each participant receives audio signal flows associated with all other avatars in his/her interactive *and* background zones. The relative position and state of the virtual environment is obtained from the state server(s). Using this information, the client device performs the appropriate spatial audio placement operation. The peer-to-peer unicast delivery architecture is shown in Fig. 3.2.

This solution will, in many cases, provide a best-case delay for the immersive audio communication environment (assuming shortest path and low congestion routes). It also has the advantage of using free or cheap computation power within the client devices and having no single point of failure. However, this architecture faces serious bandwidth inefficiency and scalability problems. If there are a large number of avatars in the interactive zone and background zone of an audio scene, a large number of peer-to-peer audio flows are required, resulting in high core and access network bandwidth usage, congestion, and scalability problems, as shown in Fig. 3.2. With

limited access bandwidth such as modem, or ADSL, assuming reasonable quality audio data rate of 16 kb/s, the peer-to-peer architecture can cause serious scalability problems in both upstream and downstream directions.

If multicasting is used at edge nodes, a game client only needs to send one audio flow, but still needs to receive all audio streams from other clients in her/his hearing range. The dynamic movement of avatars in the virtual world can cause frequent changes in multicast groups, which may make the control complexity of multicast outweigh the bandwidth cost saving.

An additional disadvantage of the peer-to-peer architecture is that security and anonymity is reduced by the fact that this method of delivery may reveal participants' IP addresses to others.

In addition, there are other variants of peer-to-peer where some peers are elected to act as partial servers for others. However, these variants of peer-to-peer are not considered in this thesis since the peers that are elected to work as partial servers for others will require higher access bandwidth. Also, when these peers leave the game, dynamic redirection of audio flows for all other peers will be required for delivering the partial audio mix.

3.3.2 Central Server

Using a central server to deliver the immersive audio scene is shown in Fig. 3.3. In order to create the audio scene, voice is streamed from each of the client devices to the central server. All voice mixing is done at the central server. The central server uses these streams in conjunction with avatar position information to create an immersive audio scene from the perspective of every avatar. An immersive audio scene may consist of multiple streams and these streams are sent back to each participant. In this scheme, the core network bandwidth usage scales linearly with respect to the number of participants and there are no scalability issues with respect to the access bandwidth. The implementation of security, privacy, billing and other policies is also possible.

The use of a central server for the provision of an immersive audio service for a

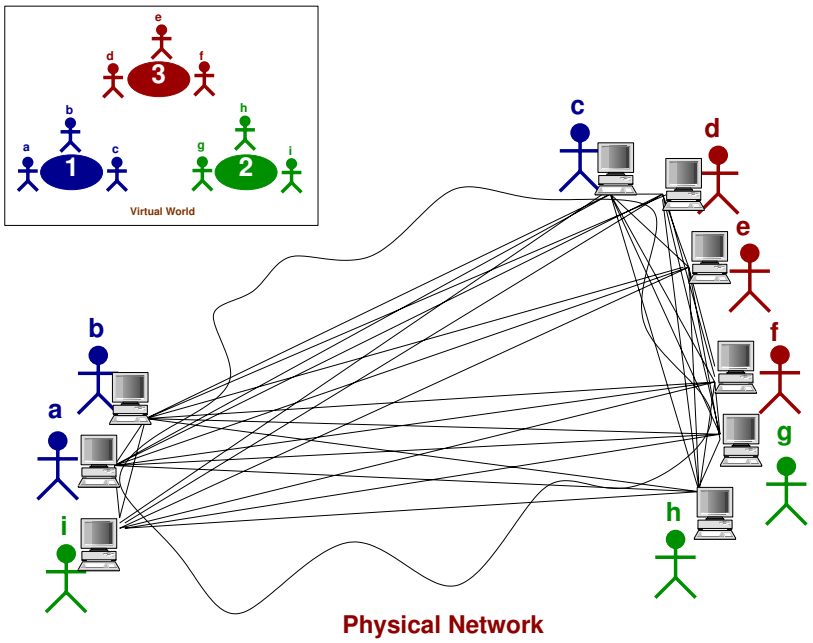


Figure 3.2 Peer-to-peer architecture for immersive audio scene creation

MMOG will introduce higher delays for voice streams since they are routed via one central server. However, this architecture has low core network bandwidth usage. The problem of access bandwidth in peer-to-peer is eliminated since each user only need to send one stream to the central server and receive a small number of streams from the server. In addition, as all audio flows are processed at a centralized server, summarization of audio flows from a group of avatars to a single source can be reused as demonstrated in (Boustead et al., 2005).

It is interesting to note that the use of a central server for state information can be tolerable since games can be written to be less tightly delay constrained than first person shooter games such as Quake and Half Life. In addition, state information delay and jitter can be compensated for, in many cases, by using techniques such as “dead reckoning” to ensure smooth graphics (Singhal and Zyda, 1999). In networked games, ”dead reckoning” is a technique that estimates a game object position based on previous positional and speed information. This technique is implemented by broadcasting the position and speed of each moving object. When the movement

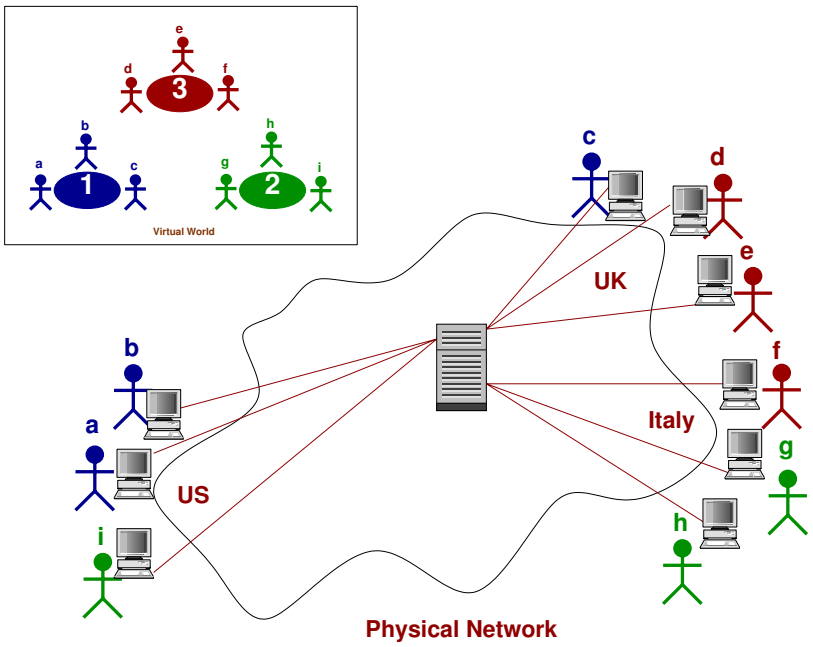


Figure 3.3 Central server architectures for immersive audio scene creation

packet is received, the game client interpolates the object's current position. This can be referred to as client-side prediction. There is also a server-side rollback technique (Bernier, 2001). In this technique, due to the latency between game players and the game server, when receiving a game event from a player, the server moves the state of the game virtual world back to the time when the game action was invoked at the player. After processing the game event, the server moves the state back to its current condition. In short, game state interactivity such as moving, shooting, scene rendering can make use of latency compensation technique. However, no such techniques exist to compensate for delays in voice streams that are used for interactive conversation.

3.3.3 Distributed Locale Servers

Fig. 3.4 shows the basic operation of the distributed locale server architecture. These servers form a server overlay network. The virtual world is partitioned into locales, and the audio mixing operations in each locale is performed by one server. It is pos-

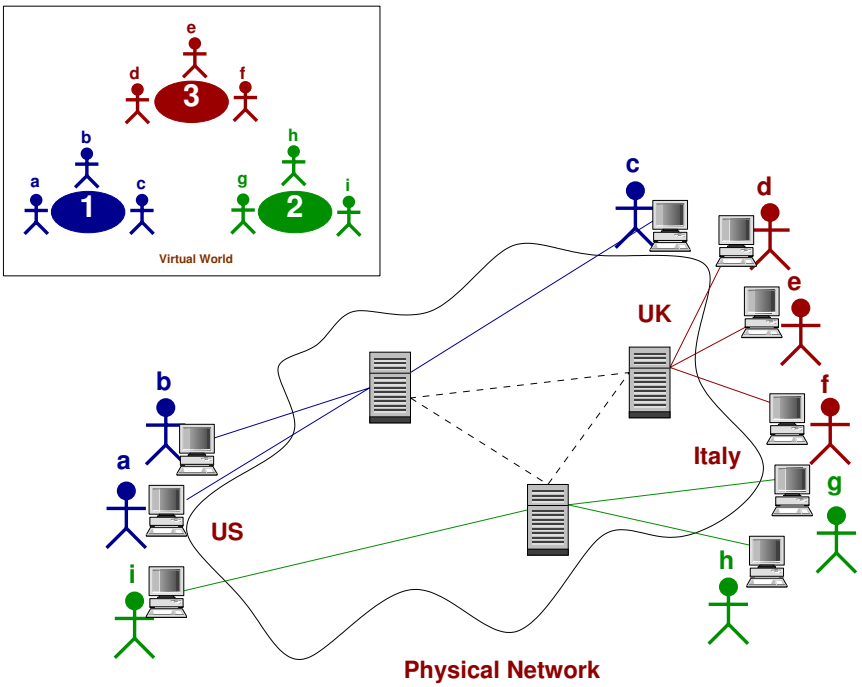


Figure 3.4 Distributed locale server architecture for immersive audio scene creation

sible to assign one server to more than one locale. Avatars in each locale only need to send their audio stream to the server assigned to that locale. If avatars in two adjacent locales, which are assigned to two different servers, are in the hearing range of each other, the two servers need to exchange the required audio streams (shown as broken lines in Fig. 3.4). When servers are located in different geographical regions, distributed locale servers may provide better delay performance compared with the central server, especially when people from a particular geographical location aggregate as a group in the virtual world. In this situation, an optimal server assignment algorithm is crucial to optimize the delay performance of this architecture. Since computations are distributed, this architecture is more scalable than the central server and has no single point of failure. Finally, similar to the central server architecture, implementation of security, privacy, and billing is also possible.

3.3.4 Distributed Proxies

Fig. 3.5 shows the basic operation of a distributed proxy architecture, in which, game players in each geographic region are connected to a close proxy. These proxies form an overlay network. These proxies also communicate with the game state server(s) to obtain the state of the virtual environment, such as avatar positions. Based on this information, these proxies manage the audio mixing operation on behalf of their clients. In particular, each proxy is responsible for receiving audio streams and mixing these streams to create an audio scene for each of its clients. They also forward audio streams to other interested proxies that need these streams for their clients' audio scenes. Similar to the peer-to-peer architecture, the proxies can send audio streams in either unicast or multicast.

The distributed proxy architecture solves the access bandwidth problem of the peer-to-peer architecture since it has a similar access bandwidth requirement to the central server architecture. In addition, security and privacy of players may be improved compared with the peer-to-peer architecture. For example, a proxy can hide a player's identity such as IP address. Once an IP address is known, it is possible for a malicious user to concentrate on the player's computer for the purpose of cracking it. In addition, IP address information can be used to make the computer a target of a Denial of Service attack.

In terms of delay, if a proxy is located at every POP, the distributed proxy architecture has a similar delay performance to the peer-to-peer architecture. This scenario is idealized and we expect that in reality, the number of proxies will be smaller than the number of POPs. Since computations are distributed, the proxy system is more scalable than the central server and has no single point of failure. If a proxy fails, audio streams to/from game players associated with this proxy can be rerouted to the next closest proxy. However, this architecture needs a very dynamic connectivity and interaction management between proxies.

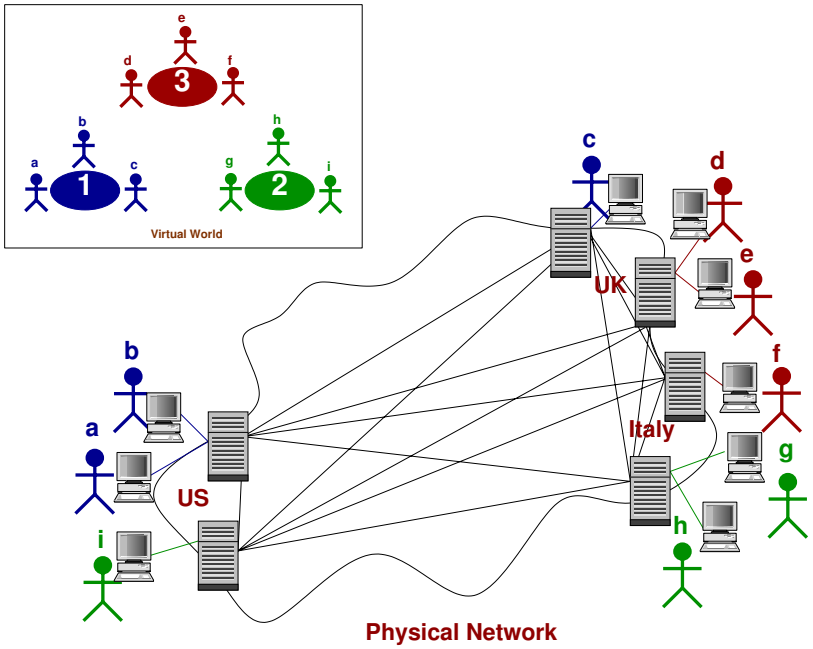


Figure 3.5 Distributed proxy server architecture for immersive audio scene creation

3.3.5 Discussion on peer-to-peer and server architectures

The peer-to-peer architecture and the server architectures have some advantages and disadvantages in terms of network bandwidth and delays. While the evaluation of network delay and core network bandwidth of the peer-to-peer architecture and the server architectures will be provided in the following chapters of the thesis, this section will discuss some interaction between server/peer-to-peer architectures in terms of current users's access bandwidth and audio codecs.

In a server architecture, a client computer only needs to send/receive up to a small number of mono voice streams. Typically, when a game client has only two speakers connected to the computer or headphone, two streams are adequate. In addition, this number can be set up at the server based on the client available access bandwidth. The disadvantage of server architectures is the additional delays as all voice streams need to be routed through servers. Depending on the network size and network delays, if the overall delay of a server architecture is below the acceptable end-to-end

delay tolerance (e.g., 150ms or 200ms), a server architecture would be acceptable.

On the other hand, the peer-to-peer architecture has low delay as voice stream are sent directly among participants. One problem with the peer-to-peer architecture is the scalability of the access bandwidth. In the peer-to-peer architecture, a participant needs to send/receive all voice streams to/from players in the hearing range. Assuming the ADPCM 32kbps codec (currently used in DICE) is used, each audio stream will require an access bandwidth of 55.3 kbps (noted in Table 2.2) in both upload and download streams. In addition, game state information also adds some traffic to the access link bandwidth. If a current ADSL user has a upload bandwidth of 256kbps, this access bandwidth will be a limitation when the number of players in the hearing range is larger than 4.

Furthermore, users often experience lower access bandwidth than the scheme offered from ISPs. For example, a measurement study from TerraCall [1] shows that with 56kbps, 64kbps, and 256kbps subscribed schemes, users normally get 32kbps, 54kbps, and 216kbps continuous data bandwidth, respectively. This will put more pressure on the access bandwidth problem of the peer-to-peer architecture.

3.4 Framework for Performance Evaluation

3.4.1 Game Player Grouping Behaviors

This section classifies differences in group formation behaviours in games in order to understand and develop the best strategies for delivery of immersive audio streams. This classification is only applicable when each participant controls only one avatar (First Person Shooters or Role Playing Games). Situations where players control many avatars (strategy games) are not considered since immersion may not be possible. As a player controls many avatars at the same time, it is difficult to identify the actual position of the player in the virtual environment in order to render the immersive voice communication

[1] ISP's, bandwidth and compression selection for your softphone, White paper, http://www.terracall.com/

Group formation behaviour is heavily game dependent as a result of different laws, rules and even game culture. Some virtual worlds encourage different types of player grouping and audio interaction. As an example, in a social virtual environment, old friends may gather in a small virtual pub for a chat. In a war game, thousands of players may gather from all around the world in a massive army. Note that, even within a given game, avatars may follow different grouping behaviours.

Audio flows need to be delivered to users in the real world based on the location of their avatars in the virtual world. Therefore, the positions in both the real and the virtual world are of concern. The avatars positions in the virtual world will determine the subsets of participants who are in the hearing range of each other. The participants's positions in the real world will affect the delays, resource cost, and routing of audio streams.

Therefore, there are two attributes that can influence the audio delivery delay and architecture.

- The number of avatars in a group: avatar density will affect resource requirements both in terms of network bandwidth and computation.
- Geographical spread of avatars within a group: players may aggregate in "cultural" groups. Since a culture (such as a language or nationality) is often linked with one or few geographical locations, some groups may have one or few preferred geographical locations of their participants.

For different types of games, we envisage that there are a range of avatar grouping patterns, referred to as *game grouping behaviors*. The following terminology is proposed to capture this range of game grouping behaviors:

- *Loners* represent isolated avatars which are relatively far away from each other and have low chance of interaction.
- A *clan* represents a medium size group, such as a hunting group, which is very common in online game. A clan typically consists of up to 20 avatars.

- A *crowd* represents a very large group, such as a stadium or a market place. A crowd typically consists of 50 avatars or more.

Fig. 3.6 summarizes the different player grouping behaviors. The horizontal axis represents the number of avatars in each group. The vertical axis represents the geographical spread of participants in a group. If participants are all coming from a small number of locations, the group is geographically close. On the other hand, if they come from all around the world, the group is geographically spread. This parameter does not really affect loners, since their audio interaction always stays limited. However it can have a very significant influence on clans and crowds in terms of audio delivery delay and required resources.

An example of a geographically spread crowd is a virtual marketplace, while a geographically close crowd could be a virtual stadium where people from the same country support their national team. A clan of friends could be geographically close if they physically know each other and live in the same region or geographically spread if they became friends on-line.

Even when there is a cultural basis for a group, the geographical spread of players can be varied. As an example, members of a German speaking group are likely to be located in Germany or in a country nearby. However, in a French speaking group, members might be located in both France and Quebec.

We do not include in this study any dynamics. More precise classification could be done by introducing group movements in the virtual world and membership alterations (joining and leaving groups). Including dynamics in this study will require an additionally substantial amount of work. In addition to avatar aggregation behaviors that are introduced in this section, avatar mobility patterns for different games need to be investigated. In order to adequately understand the effect of avatar movements, extensive simulations and modelling are required, especially with the distributed locale server architecture. In particular, in this architecture, avatar movements may cause the current server assignment to drift from optimal. Hence, to optimize the delay performance of this architecture, it would be necessary to reassign servers to locales after a certain amount of time. Avatar movements have less impact on server

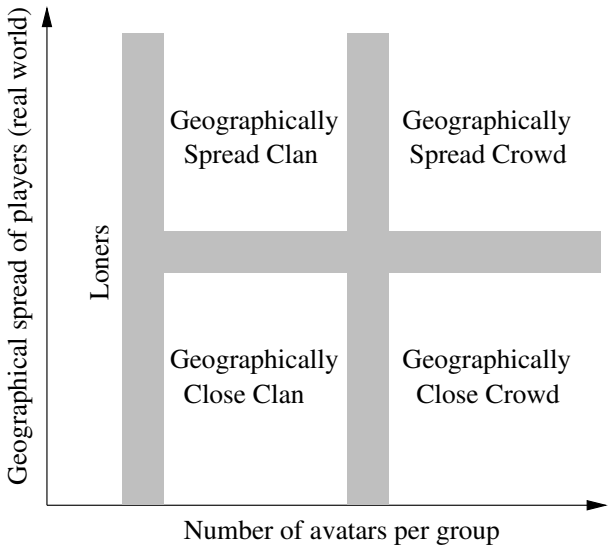


Figure 3.6 Player behavior classification

assignments in the central server and the distributed proxy architecture. Some hints of the effect of avatar mobility on the distributed locale server architecture and the distributed proxy architecture are discussed in Sections 5.2.4 and 7.2.2.

3.4.2 Models of Physical Networks and Virtual World

3.4.2.1 Physical Networks

We model the network and server topology as a graph $G_p(V_p, E_p)$; where V_p denotes a set of vertices, E_p denotes a set of edges; a set $S \subseteq V_p$ denotes a set of potential processing servers; $R \subseteq V_p$ is a set of Internet Service Provider Points of Presence (ISP POPs). In addition, R and S are disjoint sets and all nodes can support routing functions. Each ISP POP has n_i game clients connected to it. Each link has two metrics: a link cost for policy-based shortest path routing, and a link delay representing the propagation delay between the two nodes. The number of game players located at ISP POPs are randomly generated based on a uniform distribution. We do not consider the delay from ISP POPs to game players since these delays do not depend on the location of processing servers and can not be controlled by us. Also, this delay

is fixed regardless of the choice of delivery architectures and can be considered as a small offset to the delay result. A number of nearby ISP POPs are denoted as a region in the physical network.

Both unicast and multicast routing are implemented. The unicast routing is implemented using "shortest path" routing. This routing scheme is to model the operation of OSPF (Moy, 1998) which is one of the current routing protocols in the Internet. Network multicast routing is also implemented by using the shortest path algorithm. In this case, each node in the graph is a multicast node. For each avatar, a multicast group is formed comprising of people that need the audio stream from that avatar for their audio scenes.

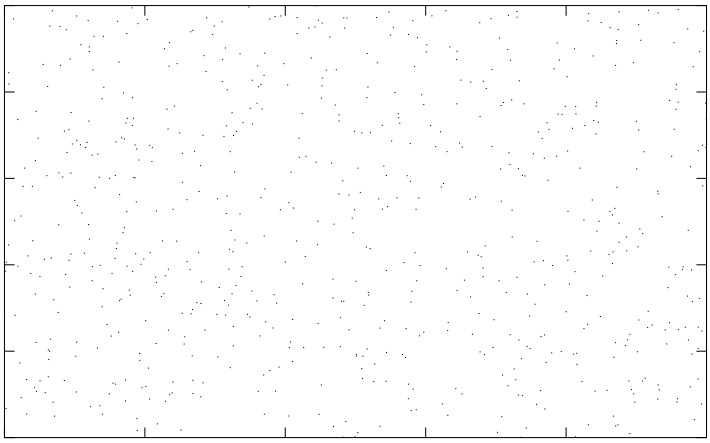
3.4.2.2 Virtual World

The virtual world is modelled as a square area of certain size, in which, avatars are distributed according to the following distributions.

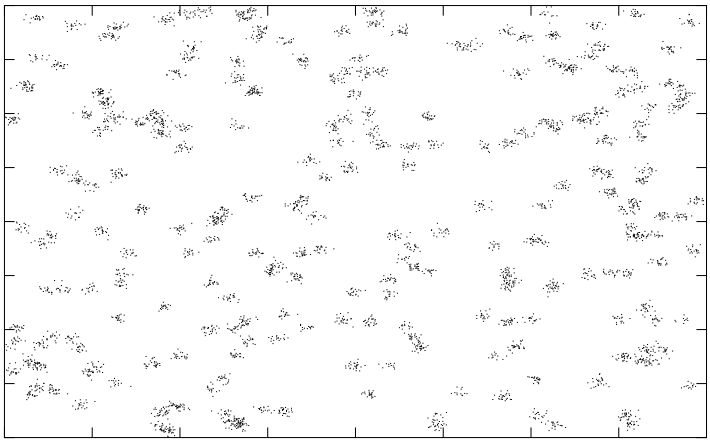
- Uniform distribution: avatar (x,y) coordinates are set according to uniform random distribution. It results in a uniform spread of avatars in the virtual world.
- Clustered distribution: Cluster centers are randomly placed in the virtual world. Avatars are positioned around these centers according to normal distribution with the mean of 0.

The above distribution is demonstrated in Fig. 3.7. In Fig. 3.7a, we model the virtual worlds which mainly have loners as a uniform avatar distribution with low density. Clans are modelled as small clusters, which consist of up to about 20 avatars. Crowds are larger clusters, which consist of about 100 avatars or more. Fig. 3.7b and Fig. 3.7c shows a virtual world which consists of 250 clans and 25 crowds, respectively.

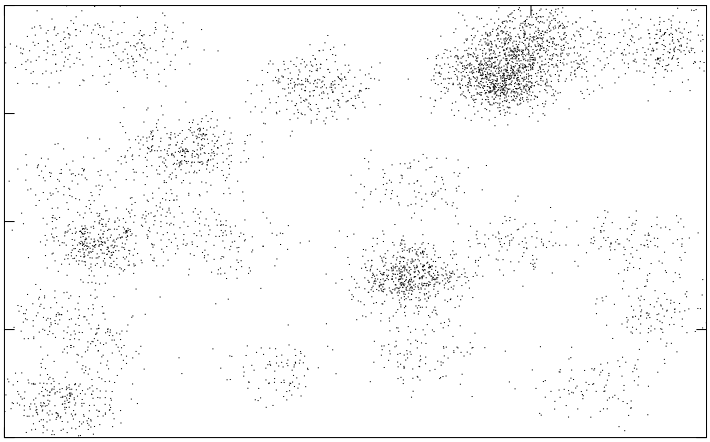
As discussed in the previous section, there may be a tendency for game players from close geographic regions to aggregate together due to language, culture, and lifestyle preferences. Two parameters loosely referred to as "correlation" are introduced to specify how people aggregate in each clan/crowd based on their real world



(a) Loners



(b) Clans



(c) Crowds

Figure 3.7 Avatar distribution in different games

geographic locations.

- The *low spread* correlation states the probability that people in each crowd/clan reside in a particular geographic region. This region is modelled as an ISP POP or a number of close POPs. This type of correlation refers to geographically close cultural groups.
- The *high spread* correlation states the probability that people in each clan/crowd are from a small number of separate regions. These regions are modelled as two random ISP POPs. This type of correlation refers to geographically spread cultural groups.

Please note that even in high spread correlation, the participants come from a small number of POPs, which are geographically spread. It is different from the uncorrelated case, where participants can be from any POP. In the performance study in the following chapters, low spread correlation is used as the default correlation parameter. High spread correlation is stated explicitly when being used.

To model these correlations, we use a correlation parameter x, $0 \leq x \leq 1$. Each time an avatar is populated in a clan/crowd, a random number r between 0 and 1 is generated. If $r > x$, the avatar is from a randomly chosen POP. If $r \leq x$, in the case of low spread correlation, the avatar is from the POP that are correlated to that clan/crowd. In the case of high spread correlation, the game player associated with that avatar is randomly assigned to either of the two POPs that are correlated with that clan/crowd. Therefore, the higher the correlation parameter is, the more people in a particular real-world geographical region group together in the same virtual location in the game.

Each avatar has an interactive zone, denoted as a circle radius R_I, and a background zone, denoted by a circle radius R_Z ($R_Z > R_I$). In our simulations in the following chapters, the interactive zone radius R_I and the background zone radius R_Z are 5m and 20m, respectively. The positions of avatars determine a subset of participants who are in the hearing range of each other. This information is used to optimize the performance of each delivery architecture.

3.4.3 Definition of Parameters and Assumption

In the evaluation of delivery architectures in the next four chapters, the following assumptions are used.

For latency evaluation, we only consider propagation delays in network links, assuming that queueing delays and transmission delays at routers are small. These delay components are indicated in Fig. 3.8. Audio mixing adds extra delay and is considered in Chapter 9. In order to measure propagation delays, approaches such as IDMap services (Francis et al., 2001) can be used to estimate distance in term of latency between nodes in the Internet. This results in an overlay network topology that consists of potential processing servers and participating routers.

For network bandwidth evaluation, we also assume that there is no congestion in the core network. Also, in server-based architectures (i.e, all other architectures except peer-to-peer), the approach in the DICE architecture (Boustead et al., 2005) is used. In DICE, a server receives one voice stream from each participant and sends up to a small number of mono voice streams to each participant (Boustead et al., 2005). In our simulation, we assume that this number is equal to 2. "2" is the minimum number of streams required for spatial voice rendering at a game client. Typically, when a player uses a headset or a two speaker sound system, two voice streams are adequate for rendering left-and-right spatial effect. Results for other numbers can be determined as the network bandwidth usage is proportional to the number of voice streams.

Two key metrics are used for the evaluation.

- **Interactive delay:** The interactive delay of each avatar is defined as the average communication delay from that avatar to all avatars in its interactive zone. As an example shown in Fig. 3.9, let d_1, d_2 and d_3 denote propagation delays of network links, the interactive delay of avatar a is calculated as the average delay from avatar a to avatar g and c. In each architecture, the interactive delay metric is defined as the average interactive delay of all avatars.
- **Bandwidth cost:** The bandwidth cost metric is calculated based on the to-

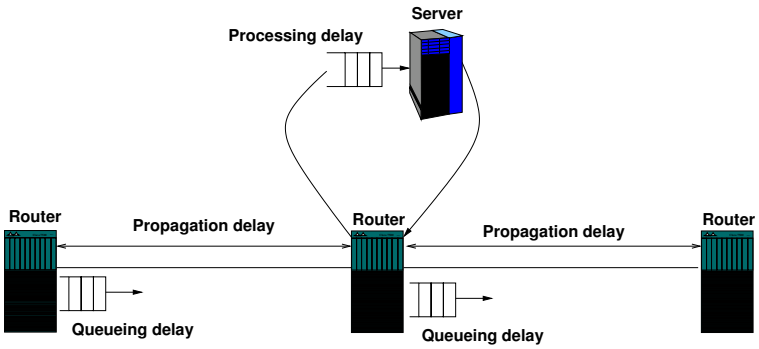


Figure 3.8 Delay components.

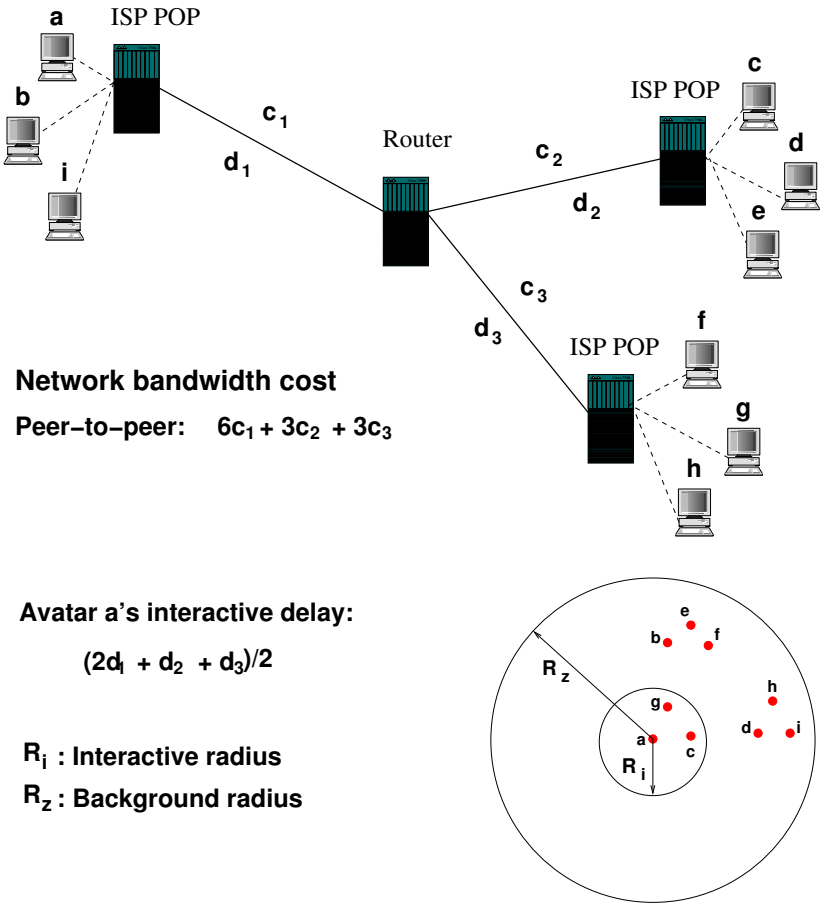


Figure 3.9 Interactive delay and bandwidth cost metrics associated with avatar "a".

tal network link costs required by all audio streams in each architecture. As an example, Fig. 3.9 shows the bandwidth cost of the peer-to-peer architecture. In the figure, let c_1, c_2 and c_3 denote the bandwidth costs of links, the network bandwidth requirements of different architectures for delivering the audio stream from avatar *a* to all other avatars is shown.

3.5 Conclusions

This chapter first introduces the concept of an immersive voice communication service for multiplayer online games. We describe several delivery architectures for this service. A simulation environment is proposed to model the service delivery. This model includes avatar distribution in the virtual world and player distribution in the Internet. Key performance evaluation parameters are also defined. In the next four chapters, we will provide an investigation of delivery architectures using the game model framework introduced in this chapter.

Chapter 4

Central Server

4.1 Introduction

The central server architecture is the easiest architecture to implement. As most of current commercial game servers use a central server, it is expected that early development of immersive voice communication service would use this architecture. When using this architecture, it is essential to choose a suitable server location to minimize latency since all communications have to be routed via the server. The research in (Chambers et al., 2003) presents a redirection service for multiplayer online games based on the geographic location of players relative to servers. Based on this service, game clients are connected to a nearby server, improving the delay performance and network bandwidth efficiency. Similarly, latency is also the key concern for the immersive voice communication service when using the central server architecture.

This chapter is organized as follows. Section 4.2 provides optimization procedures for choosing a central server from a set of potential processing servers located in different parts of the Internet. In this section, we also consider the geographic distribution of players in current multiplayer online games during a day due to time zone differences and propose a dynamic relocation of a central server. Section 4.3 provides a simulation study to evaluate the proposed solutions. Finally, Section 4.4 concludes.

4.2 Service Delivery Model

In this section, we propose two optimization procedures for the central server architecture. We use peer-to-peer as a reference to compare with our proposed delivery architectures since it provides the lowest delay.

4.2.1 Optimization Procedures

In this service model, a game service provider hires a virtual central server from a server provider (e.g., using SWON framework) to provide the immersive voice communication service for the game. Depending on the game requirements such as game player locations, the server provider will allocate an optimal server for the game. Given game player locations in the physical network and positions of avatars in the virtual world, the problem is to find an optimal central server for immersive audio scene creation. Since interactive voice communication is subject to delay between game player and the server, we propose two simple optimization objectives for placing a central server, as will be defined later.

- *Minimize average latency*.
- *Minimize interactive delay*.

The model of the physical network including ISP POPs, server sites and routers has been introduced in Section 3.4.2. Using this model, we run Shortest Path First (SPF) from each potential server s to all ISP POPs as a pre-processing step,. The distance from a server to an ISP POP node i denoted as $d(s, v_i)$, and the distance from ISP POP node i to j via the server, denoted as $d(v_i, s) + d(s, v_j)$, are pre-determined.

4.2.1.1 Minimize Average Latency

Average-latency is defined as the average delay from the central server to all game players. For each potential server s, we compute the latency from the server to each ISP POP, weighted by the number of game players at that ISP POP. We iterate through the set of potential servers and choose the server with the lowest average latency. For

each server, the average latency is defined as follows.

$$\frac{\sum_{i=1}^{N} d(s, v_i) n_i}{\sum_{i=1}^{N} n_i} \tag{4.1}$$

Where:

N: Number of ISP POPs.

n_i: Number of game players connected to POP i.

This procedure has the same complexity as finding shortest path in the graph. Since computing average-latency is based on the distance from the server to an ISP POP and the number players at that node's proximity, the central server will be placed near the centre of mass of game player population in the physical network regardless of positions of avatars in the virtual world. However, a better model is to remove isolated avatars from the delay calculation. This requires avatar positions in the virtual world and will be described in the next Section.

4.2.1.2 Minimize Interactive Delay

This solution requires both game player locations in the physical network as well as avatar positions in the virtual world. The optimization objective is to minimize the interactive delay metric. The interactive delay of player j at ISP POP i, denoted as I_{ij}, is defined as follows.

$$I_{ij} = \frac{\sum_{k=1}^{\delta_{ij}} (d(v_i, s) + d(s, v_k))}{\delta_{ij}} \tag{4.2}$$

where

δ_{ij} and v_k: Number of players in the interactive zone of player j and the set of ISP POPs that these player are connected to.

The interactive delay metric for a given server is the average interactive delay of players, defined as follows.

$$I = \frac{\sum_{i=1}^{N} \sum_{j=1}^{n_i} I_{ij}}{\sum_{i=1}^{N} \sum_{j=1}^{n_i}} \tag{4.3}$$

We compute the interactive delay metric for each server, and choose the server that provides the lowest interactive delay metric.

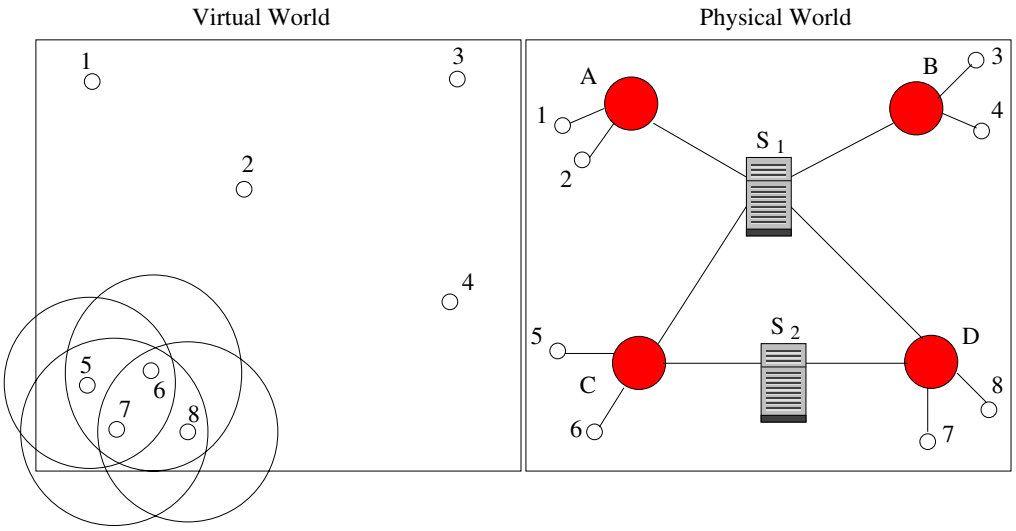


Figure 4.1 Physical/virtual world model.

The example in Fig. 4.1 shows the difference between these algorithms. In this example, the distribution of avatars is not uniform, where some are in a sparse area, others are in a dense area with lots of interactive communication. The network topology is simple and idealized in order to show the differences of these algorithms. Since *minimizing average latency* is based only on locations of game clients in the physical world, server S_1 is chosen. However, as only avatars 5, 6, 7, and 8, are in interactive communication, *minimizing interactive delay metric* chooses server S_2, instead.

4.2.2 Relocation of a Central Server

As will be shown later, it may be advantageous to change the location of a centralized server due to changes in player population distribution in the physical world or perhaps even changes in distribution of avatars in the virtual world. Time zone differences between countries can result in changes in the distribution of game participants in different parts of the network. A study in ref. (Feng and Feng, 2003) shows that the distribution of game players in different geographic regions, such as Europe, America, and Asia, has distinct peak patterns in six different 4-hour blocks during a day. In this situation, relocating the central server would achieve better delay

performance.

Naturally, if relocation of a centralized server involves hardware deployment in a different location, the time scale of relocation will be long and the cost will be significant. We assume that the game service provider can have access to a number of virtual servers located over the Internet. These virtual servers would represent some kind of potential server sites, where computation resources can be shifted to. Our primary purpose in relocating a server would be improving delay performance. Since relocating a centralized server involves some cost, we only move the server when the improvement in delay performance outweighs the cost of moving the server.

Relocating a central server may take some time, in order of seconds or minutes. The transition time includes rerouting audio flows to a new server as well as moving audio scene creation states to this server. Rerouting can be approximately equal to rerouting in overlay networks, however, the challenge is to move the state from the old server to the new server. We expect to do it seamlessly by moving the states of a small part of the virtual world to the new server step by step. During this handover period, the centralized game state server may inform game clients to direct their audio flows to the new server. These two audio processing servers exchange related information since each server would certainly have clients that communicate with the other.

The following example shows the relocation of mixing operation from one server to another during a transient period. In Fig. 4.2, suppose we have four players and the input voice streams associated with these players are denoted as $\alpha_1, \alpha_2, \alpha_3$, and α_4. The mixing operation on these input streams is presented as follows.

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} A_{11}\alpha_1 + A_{12}\alpha_2 + A_{13}\alpha_3 + A_{14}\alpha_4 \\ A_{21}\alpha_1 + A_{22}\alpha_2 + A_{23}\alpha_3 + A_{24}\alpha_4 \\ A_{31}\alpha_1 + A_{32}\alpha_2 + A_{33}\alpha_3 + A_{34}\alpha_4 \\ A_{41}\alpha_1 + A_{42}\alpha_2 + A_{43}\alpha_3 + A_{44}\alpha_4 \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$$

Where:

$[A_{ij}]$: Mixing coefficient matrix which stores the weights that apply to input voice

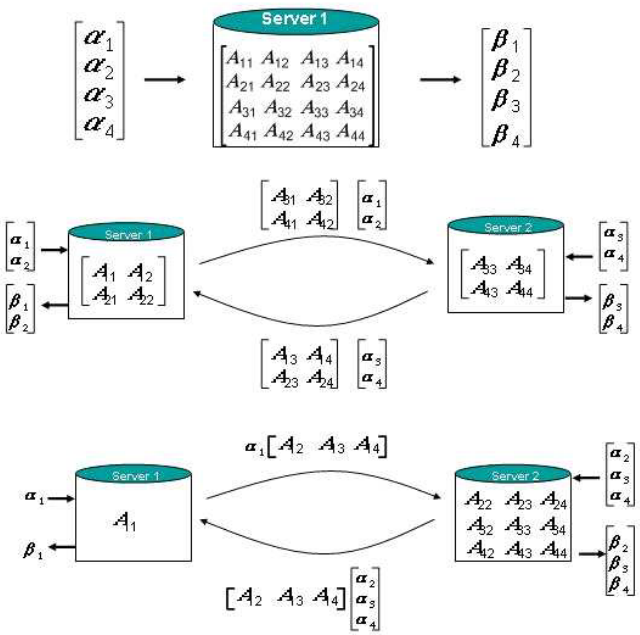


Figure 4.2 Relocation of a central server during a transient period.

streams based on position of avatars in the virtual world.

$\beta_1, \beta_2, \beta_3$, and β_4: Output voice streams.

Fig. 4.2 shows the relocation from server *1* to server *2* during a transient period. It is noted that some input streams will be forwarded to the new server and the two servers need to exchange some partial mixed streams during the transient period. This process can be described as a decomposition of the mixing coefficient matrix. In the first step, α_3 and α_4 are forwarded to server 2. The audio mixing operation in each server and exchanges between these two are presented as follows.

Server *1* receives α_1 and α_2 and provides mixing operation on these streams. This server also receives some partial mixed streams from server *2* in order to compute β_1 and β_2:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} A_{13} & A_{14} \\ A_{23} & A_{24} \end{bmatrix} \begin{bmatrix} \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

Similarly, the server *2* provides mixing operation on α_3 and α_4 and also receives some partial mixed streams from the server *1* in order to compute β_3 and β_4:

$$\begin{bmatrix} A_{33} & A_{34} \\ A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} \alpha_3 \\ \alpha_4 \end{bmatrix} + \begin{bmatrix} A_{31} & A_{32} \\ A_{41} & A_{42} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \beta_3 \\ \beta_4 \end{bmatrix}$$

In the next step, α_2 is forwarded to server *2* and server *1* only receives α_1 and computes β_1. As shown in Fig. 4.2, these two servers still need to exchange certain partial mixed voice streams. After this step, all audio mixing operation at server *1* will be completely transferred to server *2*.

4.3 Simulation Experiments

In this simulation study, we evaluate the performance of peer-to-peer, central server, and dynamic central server architectures. Firstly, we investigate the effect of changes in the geographic distribution of game players on a fixed central server in terms of network bandwidth cost and delay. We also investigate the effect of avatar distribution in the virtual world on the two proposed optimization algorithms for central servers. In addition, we compare the network bandwidth cost and the interactive delay metric of the architectures discussed earlier in a range of physical/virtual world correlation parameters.

4.3.1 Simulation Setup

We use GT-ITM topology generator [1] to model the Internet topology. Several models are included in this package such as: random graph, hierarchical, and transit-stub. We use transit-stub model since it is a better model of the Internet (Calvert et al., 1997). Our intention is to model reasonably accurate network topology and game client distributions that reflect the current MMOG delivery scenarios in the Internet. A number of potential servers and ISP POPs are chosen randomly among a set of vertices. The topology generator parameters are chosen such that the the maximum propagation delay in the shortest path between two edge nodes is 300ms. This delay is chosen based on the Internet end-to-end delay analysis in (Bovy et al., 2002). The

[1] Available at http://www.cc.gatech.edu/projects/gtitm/

Time	Peak	Mid	Off-peak
1	US	Europe	Asia
2	US	Asia	Europe
3	Asia	US	Europe
4	Asia	Europe	US
5	Europe	Asia	US
6	Europe	US	Asia

Table 4.1 Game client distribution in a day period

average node degree is between 3 and 4, which is typical for the Internet (Zegura et al., 1997). Details of simulation topology is discussed in the appendix.

4.3.2 Relocation of a Central Server

In this section, we simulate geographic changes in game player population and investigate the effect of this change on the performance of a central server over time. We use a transit-stub graph of 600 nodes, comprising of three transit domains, which reflect three main geographic regions: North America, Europe, and Asia. Each domain has on average eight transit nodes, each transit node connects to three stub Autonomous System (AS), representing the connectivity of different ASs in each region. While the current Internet topology may comprise of more than ten thousands nodes, this graph may approximate major ISP POPs in different geographical regions where game players are connected to. Details of the simulation topology are provided in Appendix B.

We randomly place 24 potential servers and 120 ISP POPs at these three regions. The number of game clients at ISP POPs in each region are varied in six four-hour blocks during a day, as shown in Table 4.1. These variations estimate the changes in game client distribution during a day, as described in (Feng and Feng, 2003).

In our simulation, the numbers of players at each ISP POP are uniformly distributed with the mean of 50 for peak, 25 for mid (the period between peak and off peak), and 5 for off-peak. Avatars are uniformly populated in a 650x650m square, and the average number of avatars in the interactive zone is 2.5 (referred as *interactive density*). At each time interval, we run the optimization algorithm to choose a new

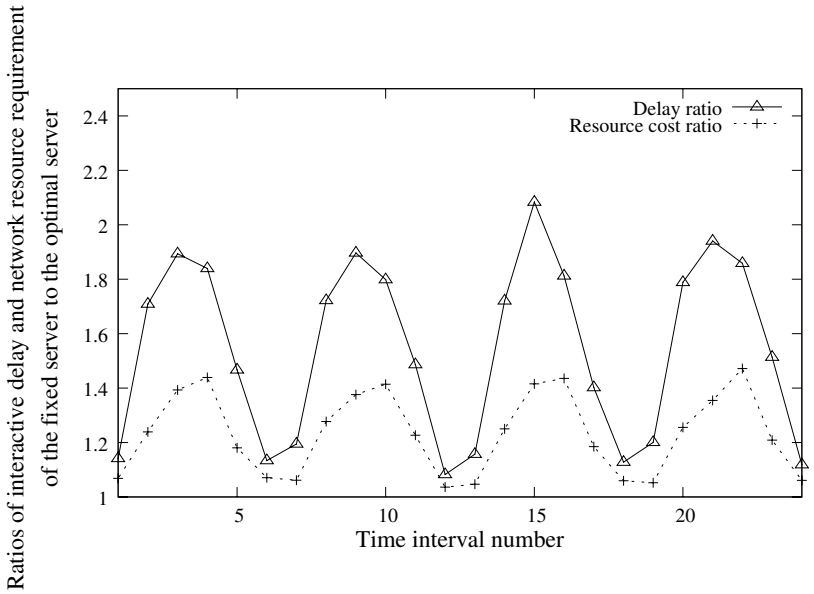


Figure 4.3 Effect of changes in game client distribution on the interactive delay metric and network bandwidth usage of a fixed central server versus the optimal central server.

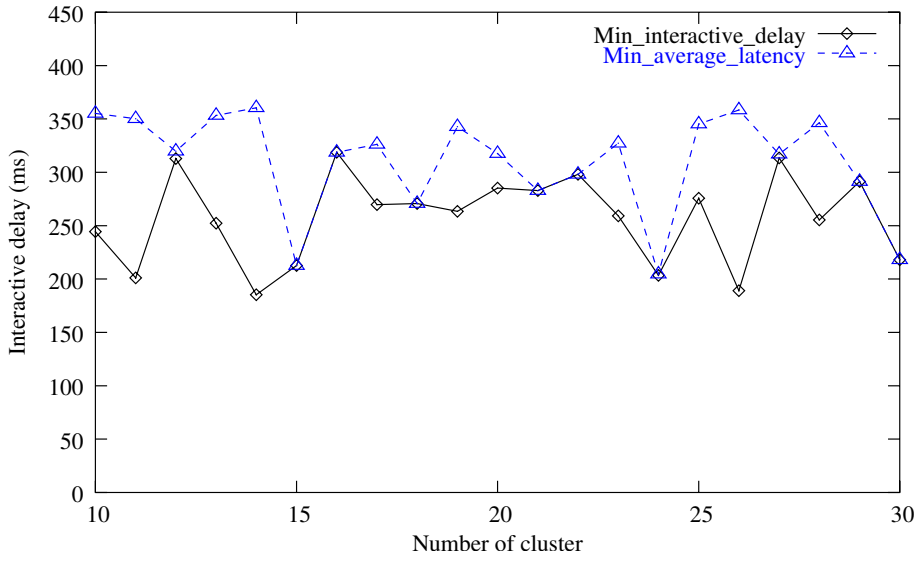


Figure 4.4 Interactive delay comparison of the two optimization objectives in different cluster distribution.

optimal central server and compare the performance of this server to the fixed server which is the optimal server at the first interval.

Fig. 4.3 shows that the ratios of network bandwidth resource cost and interactive delay of a fixed server to a dynamic optimal server in each time interval are from 1.1 to about 2.1. The periodic behaviour of the graph is as a result of the periodic distribution of game clients over time. For example, after six time intervals, the optimal server is close to the fixed server that was optimized for the first interval. The resource cost ratio is lower than the delay ratio since it is only based on the number of links rather than the propagation delays on the links. The same number of links can result in a large difference in delay. Therefore, it is more efficient to shift the fixed centralized server to a new optimal server every few hour intervals during a day. A game service provider can monitor client distribution patterns and plan server relocations with specific times during a day.

4.3.3 Comparison of Optimization Objectives

In the previous simulation, the two optimization algorithms described earlier in most case choose the same central server. It is due to a uniform distribution of avatars with high density. As a result, game clients at different ISP POPs have equal probability of being in interactive communication, and the two algorithms get the same result. *Minimize interactive delay* chooses servers with lower latency when we use cluster distribution, in which some avatars are located in a few cluster points with high interactive density, while others are in sparse area. In Fig. 4.4, we use the same graph as before but populate avatars in clusters, and the number of clusters increases from 10 to 30. Since game client distribution is not changed, *minimize average latency* chooses the same server. The result from Fig. 4.4 shows that *minimize interactive delay* always chooses an equal or a better server with equal or lower latency. In addition, the actual clustered avatar distribution in each simulation run does not have a particular affect on the delay results. The purpose of the figure is to demonstrate the advantage of *minimize interactive delay* over *minimize average latency* in different clustering cases. Therefore, depending on types of games and avatar distribution patterns (e.g, either cluster or uniform), *minimize interactive delay* can be used to obtain a lower interactive delay.

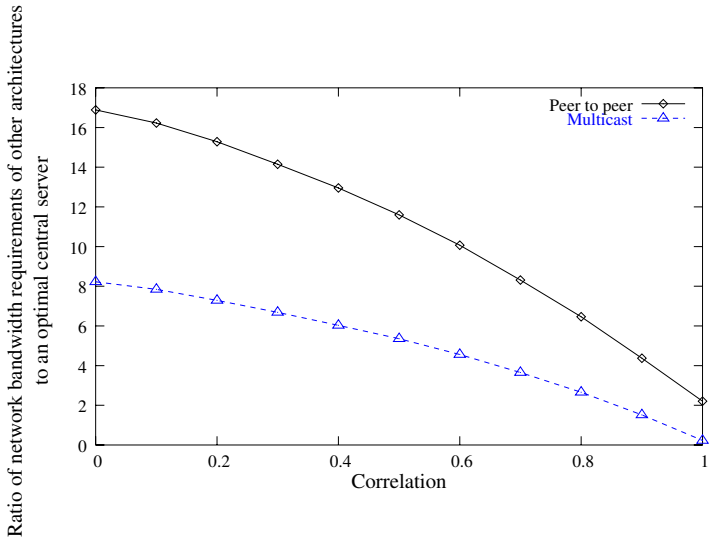


Figure 4.5 Effect of varying physical/virtual world correlation on network resource usage of multicast and peer-to-peer unicast versus the central server architecture.

4.3.4 Effect of Varying Physical/Virtual World Correlation on Network Resources and Delay Metrics

In the following simulations, the number of avatars are equal to 10,000, the average number of avatars in background zone and interactive zone are 40 and 2.5, respectively. The number of ISP POPs is 25. Also, shortest path network multicast routing is used to get results for the peer-to-peer architecture using multicast.

Figure 4.5 shows the ratios of network bandwidth resource usage of multicast and peer-to-peer to the optimal central server for a range of correlation parameters. As indicated in the figure, these ratios reduce as the correlation parameter increases. This is expected since the higher the correlation is, the fewer unicast flows are needed across the networks, the less resources are used. Specifically, the cost ratio of peer-to-peer architecture reduces from 17 at no correlation to above 2 at a correlation of 1. The cost of multicast is less than a quarter of peer-to-peer at a correlation of 1 but increases to nearly half of peer-to-peer cost at no correlation.

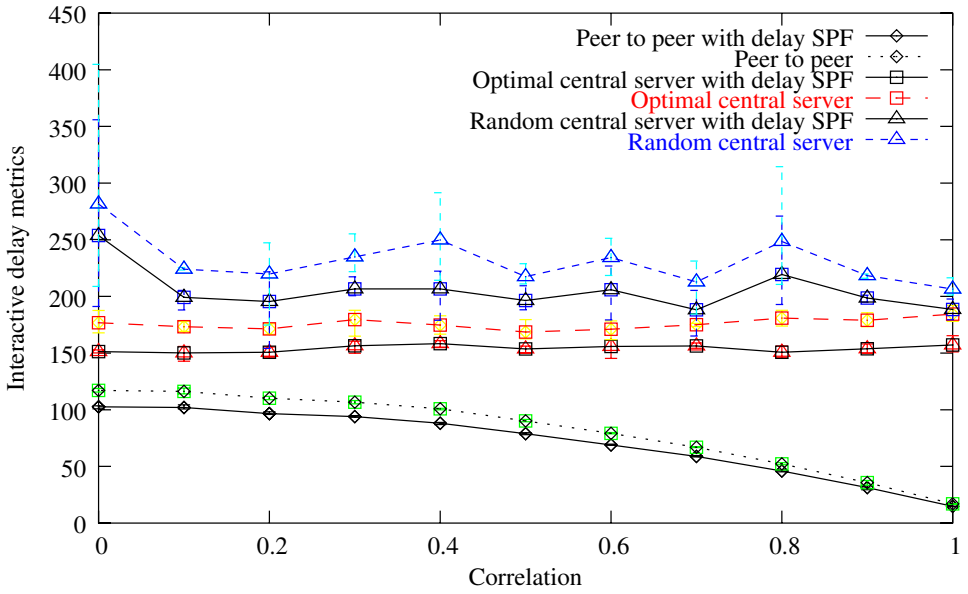


Figure 4.6 Effect of varying physical/virtual world correlation on the interactive delay metric

In Figure 4.6, the interactive delay metric of different architectures (as defined in Section 4.2.1.2) are obtained in ten simulation runs. The delay curves show the average values and error bars which indicate the maximum and minimum values of these runs. The interactive delay is calculated based on delay shortest path routing (indicated as "delay SPF" on the graph), and normal shortest path routing. In our simulation model described earlier in Chapter 3, there are two network link metrics: link cost for routing purpose, and the actual link propagation delay. The delay shortest path routing uses the actual link propagation delay as a metric to calculate the shortest path in terms of delay between two nodes. The normal shortest path routing calculates shortest path based on link cost (often being set based on hop count, policy, and bandwidth cost) which reflects the operation of some current shortest path routing protocols in the Internet, e.g, OSPF.

As shown from Figure 4.6, the interactive delay metric of an architecture using delay shortest path routing (solid lines) is about 10% smaller than that using link cost shortest path routing (broken lines). The peer-to-peer architecture has the smallest delay, and this delay increases when the correlation decreases, as expected. Delays

of centralized servers do not depend on correlation. The location of a random central server may result in high interactive delay. Especially, as indicated from the error bars, when a random server is far from the optimal server, the interactive delay metric can be more than twice the optimal server.

4.4 Conclusions

This chapter investigates the use of a central server architecture to provide an immersive voice communication service to multiplayer online games. We assume that the game service provider could hire a server from a set of potential processing sites for creating immersive audio scenes.

We provide two optimization objectives for choosing an optimal central server from a set of potential servers. One is to minimize the *average* delay from all participants to the central servers which is based only on player distribution. The other is to minimize the *interactive delay metric*, which requires both player distribution in the Internet and avatar distribution in the virtual world. In most cases, these two objectives choose the same server while minimizing interactive delay may obtain more optimal server in some avatar distribution patterns.

We propose to relocate a central server in response to changes in player distribution due to time zone differences. The simulation results show that relocation of the central server in response to these changes can significantly reduce the interactive delay by up to 40% and the network bandwidth usages by up to 50%. In addition, an optimally located central server can reduce the interactive delay of a randomly located central server by up to 50%. Peer-to-peer has the smallest delay as expected given no congestion and shortest path routing is used. However, in a dense virtual environment, the network bandwidth usage of peer-to-peer can be 17 times that of the central server architecture.

Chapter 5

Distributed Locale Server

5.1 Introduction

There are two major drivers for the distribution of applications over a network of servers: *resource driven distribution* (RDD), and *latency driven distribution* (LDD). The aim of RDD is to move components of the application to other servers to overcome the processing capacity limitations in one site. If there are sufficient resources available in one server, however, the distribution would not lead to any performance improvement. Examples of RDD are cluster computing and grid computing. For LDD applications, the ability to control the *spatial* location of the processing is of pivotal importance. So even if the processing resource is abundant in one location, the application will perform poorly in terms of response time and latency if it is not distributed over a suitable set of geographically diverse servers. The immersive voice communication service is an LDD application and we will demonstrate that the distributed locale server architecture will lead to significant improvements even when there are abundance of processing resources in each server.

The distributed locale server architecture is similar to a distributed game server architecture based on virtual location partitioning. This game server architecture has been proposed in several several research projects with the purpose of designing scalable state information servers for large multiuser virtual environments including multiplayer games in references (Barrus et al., 1996) and (Diot and Gautier, 1999). In

this architecture, a key design issue is to efficiently distribute loads among participating servers. Consequently, various research in load balancing and load sharing for distributed server architectures in DVE have been carried out in (Lui and Chan, 2002) (Ta and Zhou, 2003) and references therein. The objective of load balancing is to equalize the load among different servers. On the other hand, the objective of load sharing is to avoid situations in which some servers are overloaded while other servers are underloaded. These algorithms are proposed for large virtual environments such as MMOG.

These above research efforts in distributed game servers are based on resource driven distribution and not related to LDD aspect of immersive voice communication. The distribution of servers in this chapter is driven by latency requirements as opposed to load balancing and load sharing. By using novel server assignment algorithms, the improvement in latencies of the distributed locale server architecture is demonstrated.

In Section 5.2, we present a mathematical formulation for the optimal partitioning and server assignment and develop a heuristics approach based on a graph algorithm. In Section 5.3, a simulation study is carried out to evaluate our proposed server assignment algorithms and to discuss various factors that might improve the group communication delay. In particular, we investigate the effect of changes in the number of servers and the correlation between distribution of avatars and game participants on communication delays and network resource usages in different game scenarios. Finally, we draw conclusions in Section 5.4.

5.2 Service Delivery Model

It is assumed that a game service provider can have access to a number of servers located over the Internet. The *distributed locale servers* distribute computation load associated with the voice mixing operation by partitioning the virtual world into *locales* and assigning each locale to one of these servers. The server computes a fixed number of partially mixed audio streams for each avatar in that locale and send these streams back to that avatar. Our aim is to find the optimal way to partition the virtual world into locales and then choose the locale servers in such a way that reduces the

total delay perceived by all avatars.

5.2.1 Problem Description

We define the total interactive communication delays as *delay cost*. A key challenge is to design an efficient server assignment algorithm to minimize this cost. We partition the virtual world into N square locales. Let P be the number of potential processing servers. Each locale is processed by only one server. It is possible to assign one server to more than one locale. If avatars in two adjacent locales, which are assigned to two different servers, are in the hearing range of each other, the two servers need to exchange the required audio streams. The inter-server delays introduce additional delays for the group communication and also additional network resources. The objective is to find the optimal way to partition the virtual world into locales and then choose the locale servers in such a way that reduces the total interactive communication delay perceived by all avatars. We assume that each server has enough capacity to create audio scenes for the whole game. In another word, each server can serve infinite number of streams. The distribution of servers therefore will be solely based on latency requirements as opposed to resource limitations (LDD as opposed to RDD). Server resource limit can be considered later by putting capacity constraints in the algorithms described in the next section.

We also assume that processing times at servers are negligible. The reason for this assumption is that processing times at servers are small compared with propagation delays. In addition, our purpose is to optimize latency based on location of servers regardless of resource constraint at each server, while processing times at servers often depend on the load at these servers.

Server processing times have an insignificant impact on optimizing the latency of distributed locale servers. As we assume that each server is not overloaded, average processing times of servers in different location are small and not significantly varied. Hence, these processing times have insignificant impact on the optimization model which is mainly influenced by propagation delay due to server locations. However, comparing with the central server architecture, server processing times can result in a small delay offset when audio scenes are computed by two different servers.

5.2.2 Mathematical Programming Model

We use the following mathematical programming formulation to model the problem. Fist of all, the known parameters are defined as follows.

Let c_i^s be the sum of delays from server s to all avatars in locale i and d_{st} be the delay between server s and server t. Let m_{ij} be the number of avatars in locale i that are in interactive communication with avatars in the neighbor locale j. If locales i and j are assigned to two different servers, say s and t, the inter-server delay cost due to this assignment is denoted as k_{ij}^{st}, and $k_{ij}^{st} = m_{ij}d_{st}$.
Decision variables:

$$x_i^s = \begin{cases} 1 & \text{if server } s \text{ is chosen for locale } i \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Objective function:
Minimize:

$$\sum_{1 \le i \le N; 1 \le s \le P} x_i^s c_i^s + \sum_{1 \le i,j \le N; 1 \le s,t \le P; i \ne j; s \ne t} x_i^s x_j^t k_{ij}^{st} \tag{5.2}$$

subject to

$$\sum_{s=1}^{P} x_i^s = 1 \quad \forall i : 1 \le i \le N \tag{5.3}$$

This mathematical programming is non-linear. We use a simple method to linearize it as follows. Let y_{ij}^{st} be a binary decision variable that has the following property.

$$y_{ij}^{st} = \begin{cases} 1 & \text{if } x_i^s = x_j^t = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

The former problem is transformed to the following linear programming (LP).
Minimize:

$$\sum_{1 \le i \le N; 1 \le s \le P} x_i^s c_i^s + \sum_{1 \le i,j \le N; 1 \le s,t \le P; i \ne j; s \ne t} y_{ij}^{st} k_{ij}^{st} \tag{5.5}$$

subject to

$$\sum_{s=1}^{P} x_i^s = 1 \quad \forall i : 1 \le i \le N \tag{5.6}$$

$$y_{ij}^{st} \le x_i^s \tag{5.7}$$

$$y_{ij}^{st} \le x_j^t \tag{5.8}$$

$$x_i^s + x_j^t \leq 1 + y_{ij}^{st} \tag{5.9}$$

The constraints in equation 5.11 ensure that each locale is processed by one server. The constraints in equations 5.7, 5.8, and 5.9 are to ensure the condition in equation 5.4 is satisfied. Basically, these constraints ensure that if two adjacent locales are assigned to two different servers, the delay cost due to inter-server communication is taken into account. This mathematical formulation is an Integer Linear Programming (ILP) problem.

Since there are N locales and P servers, the number of x_i^s variables is NP. Each locale may have from three to eight adjacent locales depending on position of the locale in the virtual world. For each locale that is already assigned to a server, there are P different server assignment possibilities for each of its adjacent locales. As a result, the number of y_{ij}^{st} decision variables is $O(NP^2)$. Therefore, the total number of binary decision variables for this problem, which is the sum of x_i^s and y_{ij}^{st} variables, is $O(NP^2)$.

This problem is also NP-hard. We will prove this by making a simple assumption of parameters and show that the resulting problem is NP-hard. Let us assume that all inter-server delay costs, denoted as k_{ij}^{st}, have a value of zero. The problem can be formulated as the following LP.
Minimize:

$$\sum_{1 \leq i \leq N; 1 \leq s \leq P} x_i^s c_i^s \tag{5.10}$$

subject to

$$\sum_{s=1}^{P} x_i^s = 1 \quad \forall i : 1 \leq i \leq N \tag{5.11}$$

This problem is a basic facility location problem, which is NP-hard (West, 2001).

When the problem size is large, it is crucial to devise a heuristics to solve the problem efficiently. In the next section, we provide a greedy heuristics based on a graph algorithm.

5.2.3 Greedy Heuristic Algorithm

5.2.3.1 Graph Representation

In order to solve this problem, we propose a multi-layer graph $G = (V, E)$, where $|V| = PN$ and $V = \{v_i^s : 1 \leq i \leq N; 1 \leq s \leq P\}$. This graph has P layers corresponding to P servers, and each layer has N vertices corresponding to N locales. Each vertex v_i^s (vertex i in layer s) has a delay cost c_i^s. If locales i and j are adjacent, there is an edge connecting any pair of v_i^s and v_j^t, denoted as e_{ij}^{st}, which has delay cost k_{ij}^{st}. The problem is to find a subgraph $G' \subseteq G$ that covers N vertices corresponding to N locales and has the minimum total vertex costs and edge costs. G' should have the property that if any two vertices of G are in G', the edge connecting these vertices is also in G'.

This graph representation is similar to graph partitioning algorithms in distributed virtual environment (DVE) in (Lui and Chan, 2002) and (Lui et al., 1998). However, as these algorithms are designed for load balancing the cost of each vertex (the computation cost of each locale) is the same for any server since this cost only depends on the number of avatars in that locale. The edge cost connecting two vertices are based on the amount of information exchanges between avatars in two adjacent locales, which does not depend on location of servers assigned to these locales. If two adjacent locales are assigned to the same server, the edge cost is zero. As a result, only a single-layer graph is used.

In this work, the delay cost of each vertex depends on the spatial location of a server with respect to the physical distribution of participants in that locale. In addition, the cost of an edge connecting two vertices also depends on the servers that are assigned to each vertex. Hence, a multi-layer graph is required to capture this behaviour.

As an example, Fig. 5.1a shows a virtual world consisting of nine locales and the resulting 2-layer graph representation of the server assignment problem when two servers are used. The figure also presents the optimal server assignment solution. The vertex cost in each layer denotes the delay cost from each server to all avatars in the locale corresponding to that vertex. For example, the delay cost from server 1

to two avatars in locale L_1 is 3 while that cost associated with server 2 is 6. These delay costs are denoted by the costs of vertices v_1^1 and v_1^2 in layer 1 and layer 2, respectively. In the virtual world, the circles denote a set of avatars in adjacent locales who communicate interactively. This results in the inter-layer edge costs in the graph. The zero cost inter-layer edges are not shown to avoid cluttering the figure. In each layer, all edges have zero cost (since adjacent locales are served by the same server) and are denoted as broken lines in the figure. As shown in the figure, the costs from server 2 to all locales are higher than those from server 1 except for locales L_2, L_3, L_7. Therefore, locale L_3 and L_7 are assigned to server 2. Locale L_2 is not assigned to server 2 since the reduction in vertex cost is 1, while the additional edge cost is 2. All other locales are assigned to server 1.

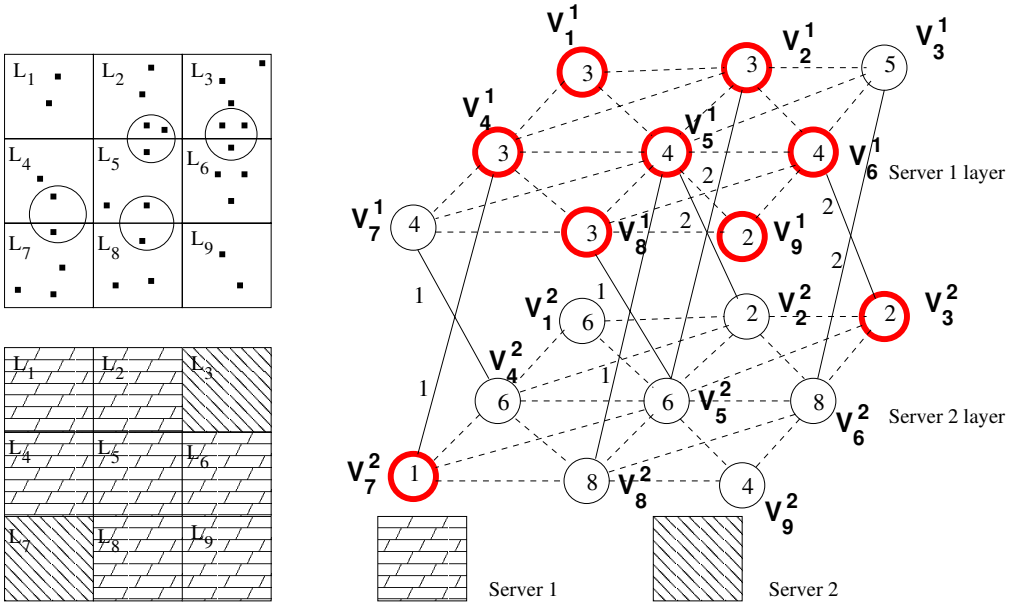
5.2.3.2 Greedy Algorithm

In this algorithm, we aim to select a sub-graph $G' \subseteq G$, that covers N vertices and has the smallest sum of vertex costs and edge costs. First of all, we choose a starting vertex for which the sum of the vertex cost and its outgoing edge costs is minimum. Then, we repeat the following procedures: choose the vertex that has the minimum sum of its vertex cost and costs of edges connecting this vertex to its adjacent vertices in the existing sub-graph. When a vertex in one layer is added to the sub-graph, the corresponding vertices in all other layers as well as all edges connecting to these vertices are deleted. This is to make sure that each locale is processed by only one server. The algorithm finishes when N vertices are covered. The run-time complexity of this algorithm is similar to Minimum Spanning Tree problem (Sedgewick, 1990). Hence, if an adjacency matrix is used for the graph representation, the run-time complexity of this algorithm is $O(|V|^2)$, where $|V| = NP$ is the number of vertices in the multi-layer graph. If an adjacency list is used for the graph representation, the run-time complexity of this algorithm is $O((|E| + |V|)log|V|)$, where E is the number of edges in the multi-layer graph.

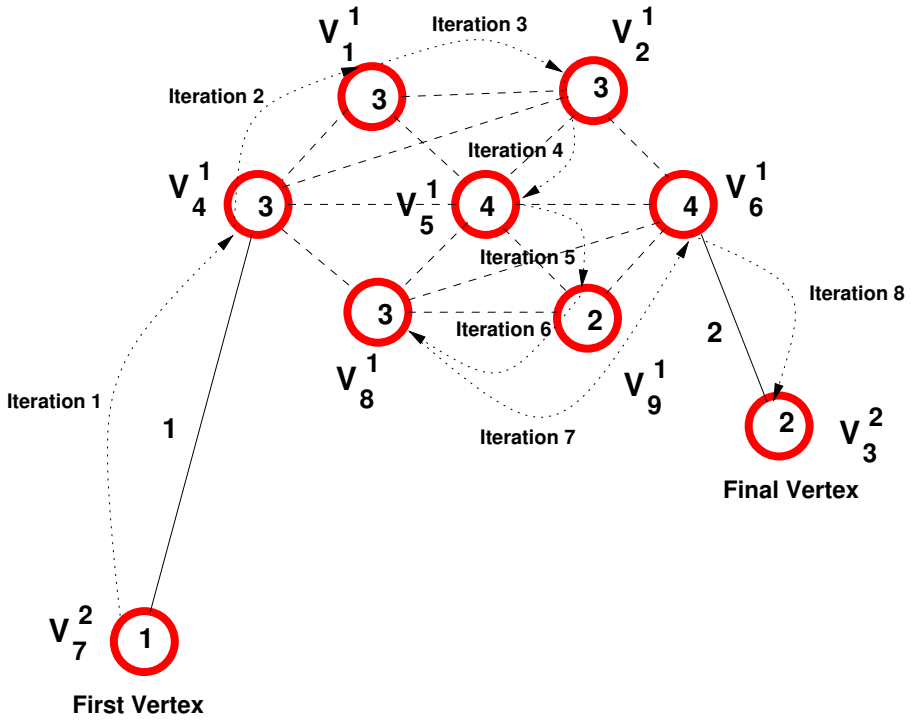
Pseudo code:

1. Initialize

$G'(V_{sub}, E_{sub}) = 0$;



(a) Graph representation and optimal server assignment



(b) Construction of the sub-graph through each iteration

Figure 5.1 Graph representation of the server assignment problem and solution.

$Neighbor(V_{nb}) = 0$;

2. Choose the first vertex v_i^s s.t ($c_i^s + \sum k_{ij}^{st}$) is minimum. Add v_i^s to G' and delete all v_i^t s.t $\{t \neq s, 1 \leq t \leq P\}$ and all edges connected to these vertices.

3. While $size(G') < N$ do

   4. Update the set of *Neighbor*: $V_{nb} = \{v_j^t\}$ s.t $\{v_j^t \notin V_{sub}$ and $\exists\, e_{ij}^{st}$ s.t $v_i^s \in V_{sub}\}$.

   5. For each $v_j^t \in V_{nb}$, compute $cost_j^t = c_j^t + \sum k_{ij}^{st}$ s.t $v_i^s \in V_{sub}$.

   6. Add v_j^t to G' s.t $cost_j^t$ is minimum.

   7. Remove v_j^t from V_{nb}, delete all v_j^u s.t $\{u \neq t, 1 \leq u \leq P\}$ and all edges connected to these vertices.

8. End

An example showing the operation of this heuristics is shown in Fig. 5.1. The algorithm first chooses vertex v_7^2 (vertex corresponding to locale L_7 and server 2) since the sum of this vertex cost and its outgoing edge costs is the minimum (this vertex cost is 1 and its outgoing edge costs are 1). Vertex v_7^2 is then added to the sub-graph G'. Also, vertex v_7^1 and all edges connected to this vertex are deleted. After vertex v_7^2 is initially chosen, the step-by-step construction of the sub-graph G' through each iteration of the algorithm is shown in Fig. 5.1b. The first two iterations are described as follows.

- *Iteration 1:* A set of neighbors V_{nb} consists of vertices that are adjacent to v_7^2: v_4^1, v_5^1, v_8^1, v_4^2, v_5^2, and v_8^1. Among these vertices, v_4^1 is added to G' since the sum of this vertex cost and the cost of the edge connecting this vertex to v_7^2 is minimum. Vertex v_4^2 and all edges connected to this vertex are removed from the graph.

- *Iteration 2:* V_{nb} consists of vertices that are adjacent to either of v_7^2 or v_4^1: v_1^1, v_2^1, v_5^1, v_8^1, v_1^2, v_2^2, v_5^2, v_8^2. Among these vertices, v_1^1 is added to G' since the sum of this vertex cost and the costs of all edges connecting this vertex to v_7^2 and v_4^1 is minimum. Vertex v_1^2 and all edges connected to this vertex are removed.

In the following iterations, v_2^1, v_5^1, v_9^1, v_8^1, v_6^1, and finally, v_3^2 are added to G'. The vertex costs and edge costs of the final sub-graph represent the total delay cost of

that server assignment solution.

If we need to consider the maximum load of each server, additional constraints are required for the ILP model. For the greedy heuristics, a constraint can be put in step 4 of the algorithm. A vertex is added to the *Neighbor* set only when the server corresponding to that vertex is not overloaded. In addition, the server load is incremented each time a vertex within the associated layer of that server is added to the sub-graph G' in step 6.

5.2.4 Impact of Avatar Movements and Player Distribution on Optimal Server Assignment

Movements of avatars in the virtual world change the composition of locales. This means that as time goes by, the current assignments of locales to servers drifts from optimal. In this situation, re-assignment of locales to servers are necessary to reduce the total delay cost.

While avatar movements may occur constantly, joining and leaving the game by participants tend to occur on a longer time scale. Especially, game players in diverse geographical regions tend to join and leave the game in different times during a day (Feng and Feng, 2003) due to time zone differences. In addition to avatar movements in the virtual world, this behavior may also affect optimal server allocation. Changes in geographic-time distribution of players may require a new assignment of distributed locale servers. However, we expect that the impact of this distribution on the distributed locale servers is not significant to that of the central server architecture.

5.3 Simulation Experiments

5.3.1 Simulation Setup

GT-ITM topology generator [1] is used to model the Internet topology. Specifically, we use a transit-stub graph of 600 nodes, comprising of three transit domains, which

[1] Available at http://www.cc.gatech.edu/projects/gtitm/

reflect three main geographic regions: North America, Europe, and Asia. Each domain has on average eight transit nodes, each transit node connects to three stub Autonomous Systems (AS), representing the connectivity of different ASs in each region. We randomly place 24 potential servers and 100 ISP POPs in these three regions. Each ISP POP has a uniform random number of game clients connected.

Unless otherwise stated, the simulation results are generated with an average of 50 game players per POP. The topology generator parameters are chosen such that the maximum propagation delay in the shortest path between two nodes is 300ms. The average number of avatars in each avatar's interactive zone in crowd/clan based virtual world is 2.5. The number of crowds/clans in a crowd/clan based virtual world is 50 and 250, respectively.

5.3.2 Investigation of Server Assignment Algorithms

In the following simulations, we compare our proposed greedy heuristics with the optimal results from the ILP model. The optimal results are obtained using Cplex optimization software [2]. The heuristic algorithm is implemented in C++ and an adjacency matrix is used for the multi-layer graph representation. These algorithms are run on a Pentium IV 2Ghz Linux server with 1.5GB of RAM. We partition the virtual world into different numbers of locales, run the two server assignment algorithms, and calculate the total interactive communication delays for both cluster (25 crowds) and uniform distributions. In these simulations we assume the correlation parameter to be equal to 0.5. In uniform distribution, we divide the virtual world into 25 squares, and the correlation parameter specifies how people aggregate in each of these squares based on their real-world geographic locations. The number of potential processing servers is 16.

As shown in Table 5.1, the results from the greedy heuristics are within 5% of optimal. However, the greedy heuristic has significantly lower running times. In particular, the run-time complexity of the greedy heuristics can be up to a hundred times lower than the ILP model. Hence, the ILP model may not be suitable for large problems.

[2]Cplex optimization software, available at http://www.ilog.com

Number of locales	Optimal	Time	Greedy	Time	Gap(%)
16	371733	0.47(s)	372066	< 0.01(s)	0.09
36	355418	1.32(s)	355590	0.05(s)	0.05
64	340376	0.74(s)	340544	0.17(s)	0.05
100	324588	3.8(s)	324588	0.45(s)	0
400	321744	670(s)	324075	8.94(s)	0.72
900	313041	30h24m	320773	70.1(s)	2.47
1225	n/a	n/a	316496	147(s)	n/a

(a) Uniform

Number of locales	Optimal	Time	Greedy	Time	Gap(%)
16	375160	0.23(s)	375160	< 0.01(s)	0
36	360761	0.46(s)	360852	0.04(s)	0.04
64	360392	0.60(s)	360542	0.17(s)	0.04
100	356572	5(s)	357042	0.43(s)	0.13
400	346074	77(s)	349397	7.88(s)	0.9
900	337340	3940(s)	343556	56.1(s)	1.8
1225	n/a	n/a	337420	114(s)	n/a

(b) Cluster

Table 5.1
Comparison between optimal results and greedy heuristic results in the total interactive communication delay.

Fig. 5.2 shows the close match between the two algorithms in a uniform distribution virtual world partitioned into 64 locales and a cluster distribution virtual world partitioned into 100 locales. Each square represents a locale and the number in each square represents the server number that is assigned to that locale. The difference between the greedy heuristics and optimal result is highlighted by shading locales for which the greedy heuristics has provided a different result.

It is also shown that increasing the number of locales reduces the delay cost. As indicated in Table 5.1, the delay cost is reduced by nearly 20% in uniform distribution and about 10% in cluster distribution when the number of locales is increased from 16 to 900. This is expected since reducing the size of locale would improve granularity of assigning servers based on the delay requirement of each avatar.

4	13	5	9	9	3	9	2
13	13	13	9	9	13	7	7
13	11	14	9	13	11	7	2
7	7	7	7	7	7	7	7
7	7	1	7	7	7	7	7
6	7	10	0	0	8	10	10
8	8	10	10	10	10	10	10
8	8	8	10	10	10	10	10

Greedy heuristics

4	13	5	9	9	3	9	2
13	13	13	9	9	13	7	2
13	11	14	9	13	11	7	2
7	7	7	7	7	7	7	7
7	7	1	7	7	7	7	7
6	7	10	0	0	10	10	10
8	8	10	10	10	10	10	10
8	8	8	10	10	10	10	10

Optimal result from Cplex

(a) Uniform

2	7	7	7	0	0	0	2	14	14
2	7	7	7	7	0	0	2	14	14
10	2	7	7	1	1	1	10	10	14
2	2	2	6	2	4	10	10	10	4
2	2	6	6	12	5	5	10	10	10
10	10	10	0	5	5	5	10	4	2
10	10	10	3	0	10	10	2	2	0
13	13	10	10	3	10	10	10	0	0
10	2	2	2	2	10	10	7	7	7
10	10	2	2	2	0	5	7	7	8

Greedy heuristics

0	7	7	7	0	0	0	2	14	14
2	7	7	7	7	0	0	2	14	14
10	2	7	7	1	1	1	10	10	14
2	2	2	6	2	4	10	10	10	4
2	2	6	6	2	5	5	10	10	10
10	10	2	0	5	5	5	10	4	2
10	10	10	3	0	10	10	2	10	0
13	13	10	10	3	10	10	10	0	0
10	2	2	2	2	10	10	7	7	7
10	10	2	2	2	0	5	7	7	8

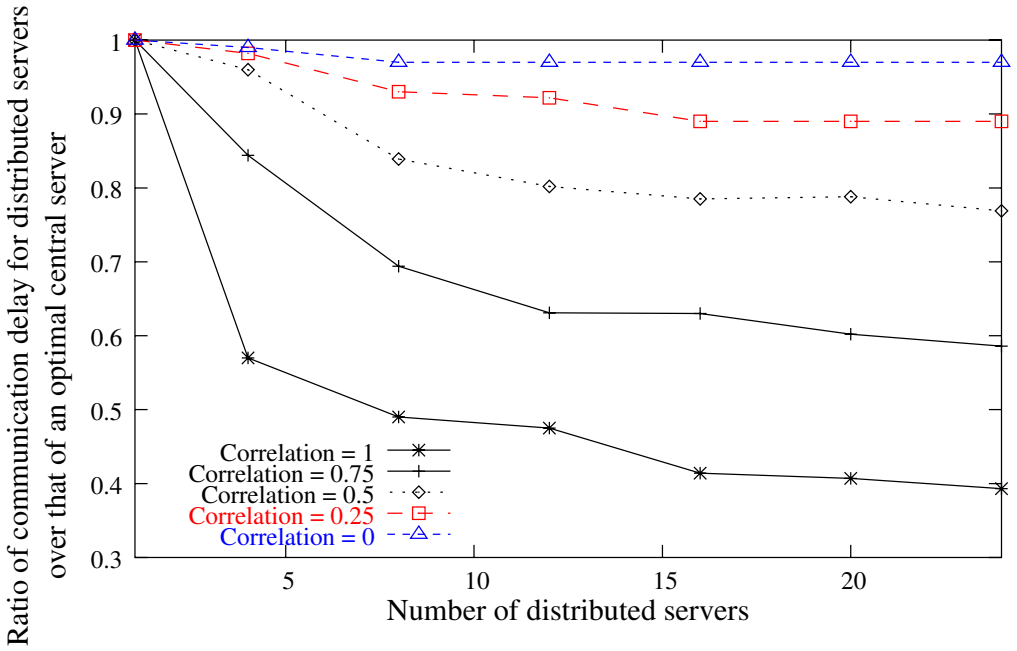
Optimal result from Cplex

(a) Cluster

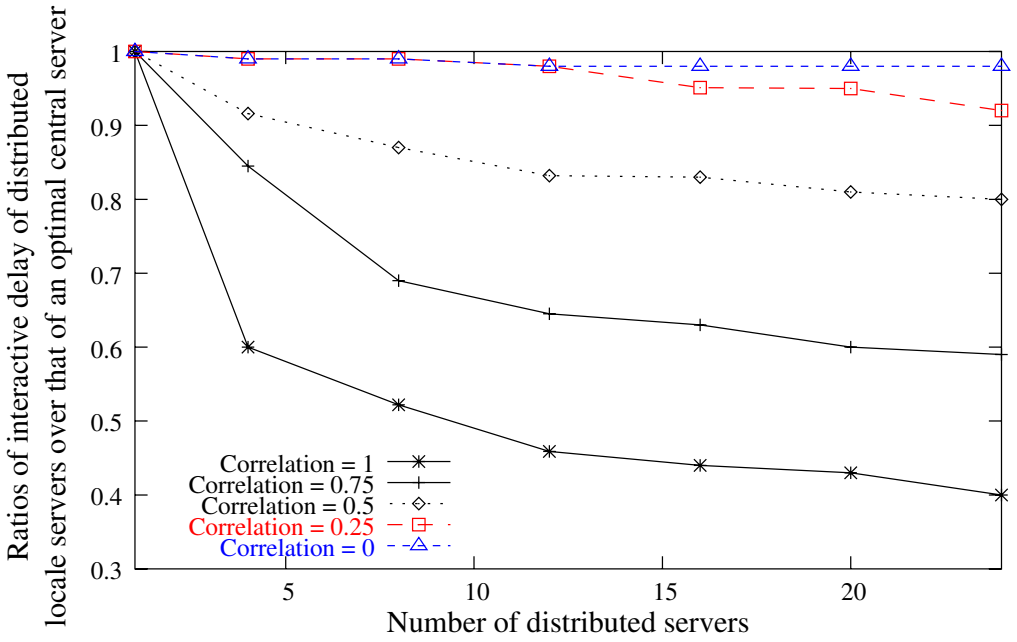
Figure 5.2 Server assignment results from Cplex and the greedy heuristics.

5.3.3 Effect of Varying Number of Servers and Physical/Virtual World Correlation

In this experiment, we investigate the effect of varying the ratio of the number of distributed servers to the number of POPs and virtual/physical world correlation on interactive delay. We compare the communication delay when increasing the number of distributed servers to that of using an optimal central server. An optimal central server is chosen among 24 potential servers. The delay cost of the optimal central server is the sum of propagation delays from all game players to that server. As



(a) Loner based distribution



(b) Crowd based distribution

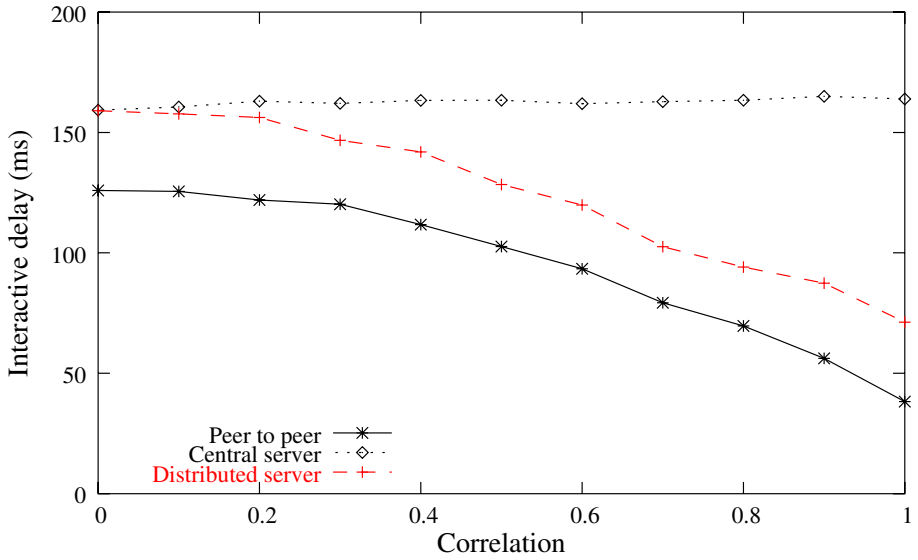
Figure 5.3 Effect of changes in number of server and physical/virtual world correlation.

shown in Fig. 5.3, at high correlation parameters, increasing the number of distributed servers would reduce avatar communication delays considerably. When all 24 servers are used, the delay cost of the distributed server architecture is reduced by approximately 60% and 20% at a correlation value of 1 and 0.5, respectively. However, at low correlation, this improvement is less significant. In addition, when the number of servers goes above certain value, approximately one eighth of the number of POPs, further improvements in delay costs are very small. This result can be taken into consideration in a real deployment scenario. Deploying a large number of server sites will result in a significantly high cost. Hence, a service provider may choose a limited number of server sites based on the number of POPs that the majority of players are connected to.

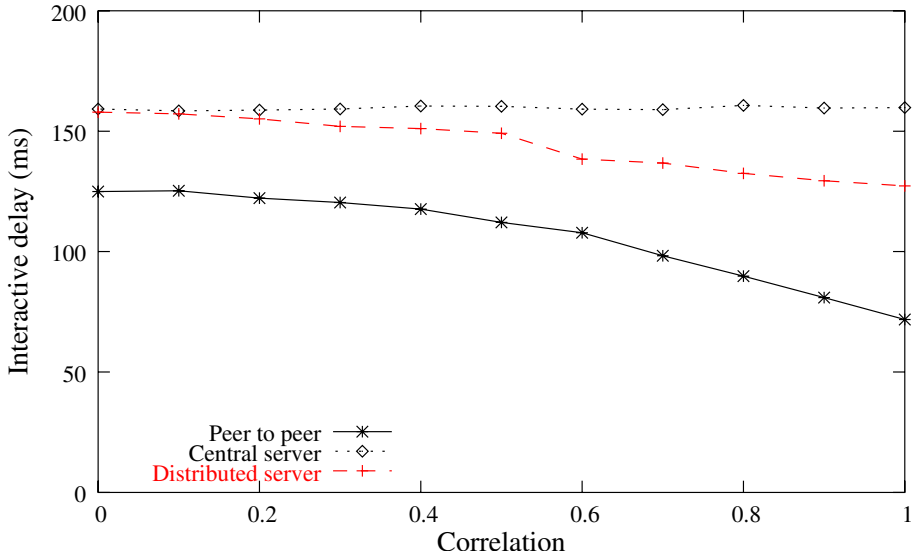
5.3.4 Effect of Varying Correlation in Interactive Delay

In the following experiments, the effect of changes in physical/virtual world correlation on the interactive delays and network bandwidth requirements of the distributed locale server architecture is investigated. Simulations were carried out to evaluate the performance of peer-to-peer, central server, and distributed locale server architecture under different avatar grouping behaviors. In the delay performance comparison, the peer-to-peer architecture with no network congestion, which has the lowest delay, is used as a benchmark. In the network bandwidth resource usage comparison, the central server architecture is used as a benchmark.

In crowd/clan based virtual world simulation, we denote low (high) spread crowds or clans if the low (high) spread correlation is used respectively. As shown in Fig. 5.4, in crowd based games, the interactive delay of the centralized server architecture does not depend on correlation, while the interactive delay of the distributed locale server architecture is reduced when the correlation increases. For the low spread correlation, the interactive delay of the distributed architecture is reduced by about 60% and 20% at correlation of 1 and 0.5, respectively, and it closely follows the "best case" peer-to-peer delay. For the high spread correlation, the delay improvement of the distributed architecture is less than that of the low spread type due to a large physical distance between any two random POPs, which have participants in that

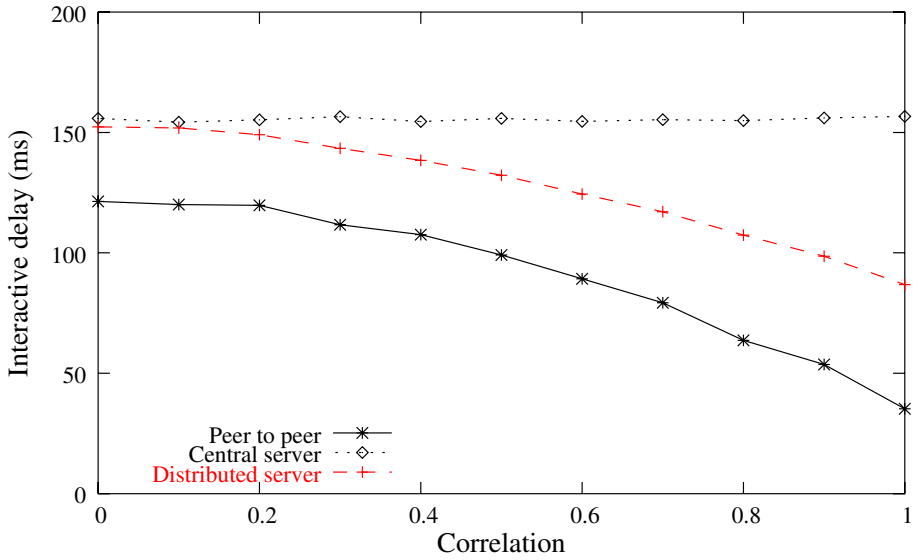


(a) Low spread crowds

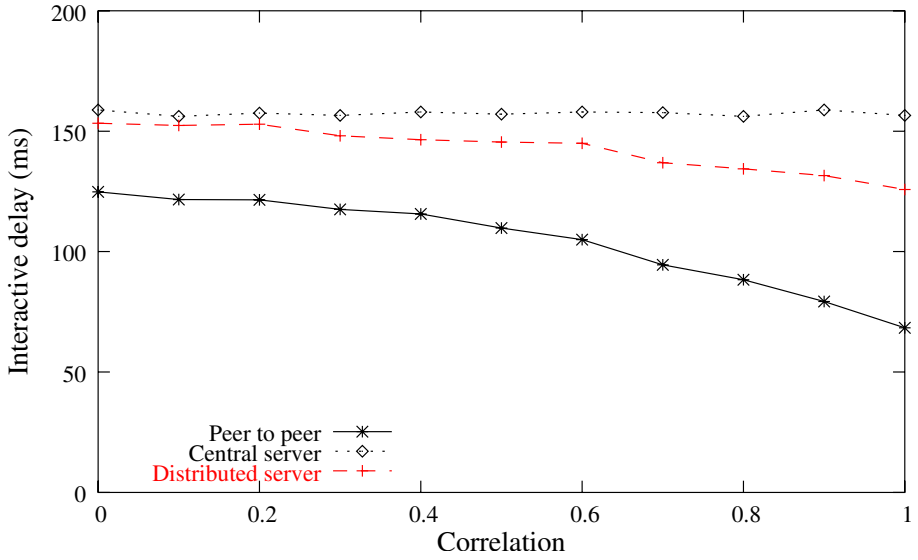


(b) High spread crowds

Figure 5.4 Effect of changes in correlation on interactive delay for crowd based games.



(a) Low spread clans



(b) High spread clans

Figure 5.5 Effect of changes in correlation on interactive delay for clan based games.

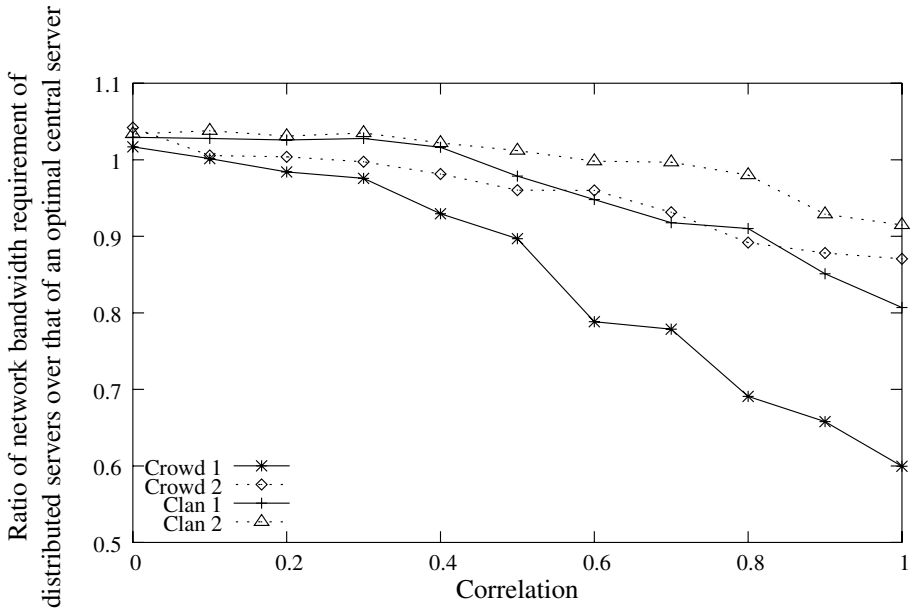
crowd. A similar behaviour is shown for clan scenarios in Fig. 5.5.

5.3.5 Network Bandwidth Requirements in Different Avatar Aggregation Behaviors

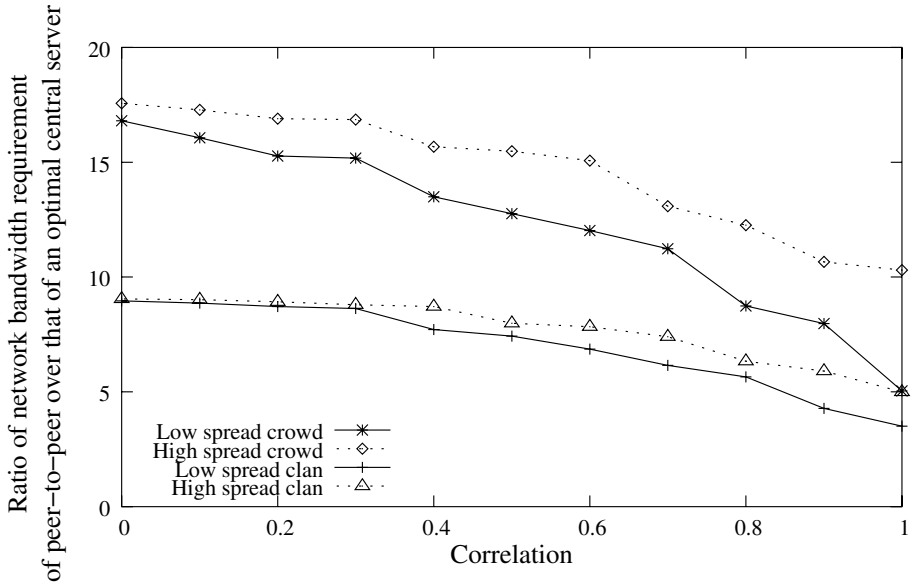
The following simulations show the network bandwidth requirements based on different avatar grouping behaviours. The network bandwidth usage of the central server architecture is used as a benchmark.

Fig. 5.6 shows the bandwidth requirement ratios of the distributed server architecture and peer-to-peer architecture versus the central server architecture for both crowd based and clan based games with respect to the two types of correlation. It is shown that the low spread correlation case results in smaller network resource usage than the high spread correlation, as expected. With the low spread correlation, the bandwidth requirement of the distributed server architecture is nearly 40% and 20% below that of the centralized server architecture in a crowd based virtual world and clan based virtual world at a correlation of 1, respectively. It is noted from Fig. 5.6b that crowds require nearly twice the bandwidth resource of clans due to larger number of avatars in each avatar's hearing range. In addition, even at the highest correlations, the network bandwidth requirements for both crowds and clans in the peer-to-peer architecture are still above three times that of the centralized server architecture.

The impact of avatar density on network bandwidth requirements is shown in Fig. 5.7. In this experiment, we simulate a loner based virtual world, in which, the average number of avatars in hearing ranges is varied from 0.5 to 5, and the correlation parameter is equal to 0. As indicated in the figure, the ratio of network bandwidth requirement of the peer-to-peer architecture over the central server architecture increases linearly from under 0.5 to about 2. The network bandwidth requirement of the distributed locale server architecture is approximately equal to that of the central server architecture. In short, in loner based game, the bandwidth usage of the peer-to-peer architecture is low and not much different from the other architectures.



(a) Distributed server



(b) Peer-to-peer

Figure 5.6 Bandwidth resource requirements of distributed server and peer-to-peer versus central server for crowd/clan based games

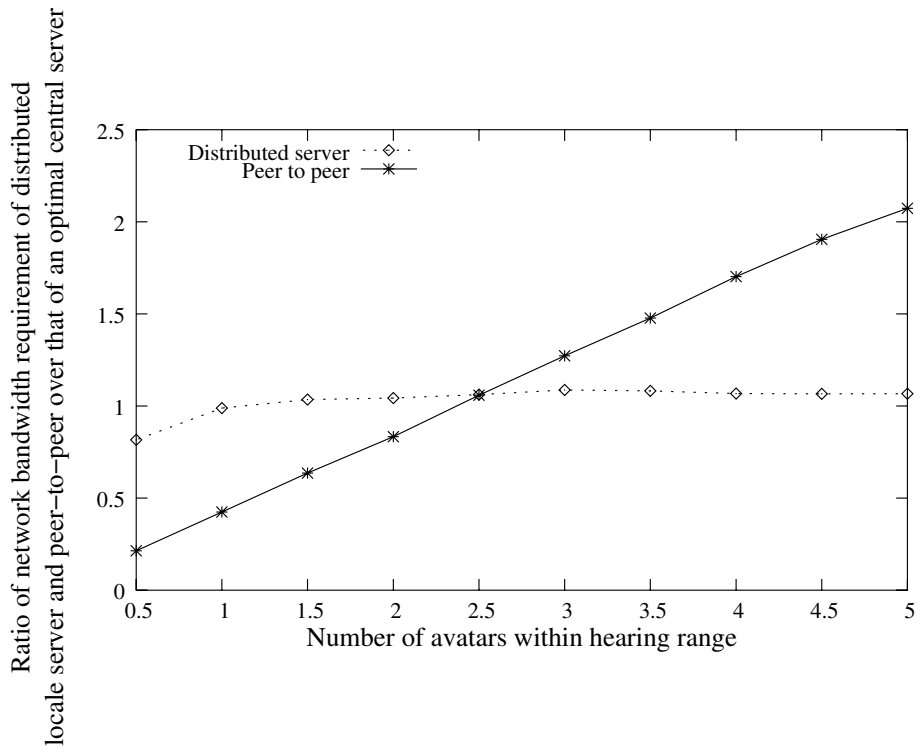


Figure 5.7 Network bandwidth requirements in a loner based game

5.4 Conclusions

In this chapter, we propose a distributed server architecture to improve the communication delay perceived by users and increase scalability of an immersive voice communication service. Two algorithms are proposed to optimize the performance of this architecture. We first formulate the server assignment problem as an integer linear program. This algorithm provides an exact solution to the problem but would have high computational complexity when the problem size increases. Hence, we produce a multi-layer graph representation of the problem. Based on this graph, we devise a greedy heuristics which has comparable performance to the optimal algorithm and low computation complexity. In particular, the results from the greedy heuristics are within 5% of optimal in all cases examined in our investigation.

We present the result of our analysis and simulation and discuss various factors that might improve the group audio communication delay. In particular, we investigate the effect of changes in the number of servers and the correlation between distribution of avatars and game participants on communication delays and network resource usage. From a simulation study of different avatar aggregation behaviours, it is demonstrated that our distributed locale server architecture can reduce the overall delay by around 20% compared to an optimally located central server and can have lower network bandwidth usage than the central server architecture. Also, increasing number of servers would enhance the delay performance of the distributed locale server compared with the central server. However, when the number of servers goes above certain values, approximately one fifth the number of POPs, further improvements in the delays are very small. In addition, the interactive delay is reduced when there is more correlation between distribution of avatar and game participants.

The avatar density in games does not affect the performance of central and distributed locale servers but has a strong impact on network bandwidth usage in the peer-to-peer architecture. In crowd/clan based virtual world, the network bandwidth usage of peer-to-peer are considerably larger than the central server and distributed locale server architectures. On the other hand, in loner based virtual worlds, the network bandwidth usage of peer-to-peer is low and comparable to the central server architecture.

Although peer-to-peer has lowest delay, its core and access bandwidth requirement is a critical limitation in parts of the virtual world with clans and crowds. In addition, there is a trade off between network bandwidth usage and latency in these architectures. While peer-to-peer results in lowest delay but highest network bandwidth usage, distributed locale server compromises between these two. Distributed locale servers outperform the central server in both latency and network bandwidth usages, especially when there is virtual world/physical network correlation. These factors should be taken into account in the design of a cost effective delivery architectures for different games.

Chapter 6

Distributed Proxy Architecture

6.1 Introduction

This chapter investigates the use of distributed proxies for providing an immersive voice communication service for massively multiplayer online games. Distributed proxies solve the access bandwidth problem of the peer-to-peer architecture while still maintaining low latencies. Peer-to-peer can be considered as a special case of distributed proxies where each proxy is placed at a game client computer. In this chapter, to address limited access bandwidth problem of the peer-to-peer architecture, we envisage that proxy servers are placed at network edges and have high bandwidth connections to the Internet.

This architecture is related to distributed game server using physical partitioning (mirrored server architectures) and distributed game proxies, which are discussed in Section 2.3.1. In all of these architectures, game clients are connected to the closest server. However, there are differences in the way information is exchanged between servers.

In the mirrored server architectures, each server has to maintain a copy of the game state and dynamically exchange messages to maintain the game state consistency. On the other hand, in the the case of immersive audio processing, each proxy server only needs to forward audio streams to other interested proxies based on avatar positions. These interactions are commonly less dynamic than those in the mirrored

game server architectures.

In distributed game proxies, each proxy does not need to maintain a copy of the whole game state. Instead, each proxy only process some functionality, as delegated from the central server. Hence, this architecture is more closely related to the distributed proxy used for immersive audio processing described in this chapter. A distributed game proxy system in (Mauve et al., 2002) has similar advantages to the architecture presented in this chapter, including robustness, minimizing network delays and privacy.

To minimize communication latency in the game, a key issue is provisioning of proxy servers. There have been various research studies in optimal placement of proxy caches and other proxy services in (Qiu et al., 2001) (Cronin et al., 2002b) (Choi and Shavitt, 2003) and references therein. A specific study on the provision of proxy servers for an immersive voice communication service is essential.

In either the peer-to-peer or distributed proxy architecture, one of the key issues is to define groups of avatars in hearing range and update the change of these groups due to avatar movements. This requires dissemination of avatar grouping information to each peer or proxy. This information is then used to determine how audio streams are sent to other proxies. The dissemination techniques are discussed in (Zou et al., 2001) and references therein.

In addition, transmissions of voice streams can be either performed by unicast or multicast. Multicast has been used for transmitting state information in multiplayer online games in (Diot and Gautier, 1999) and references therein. A general study of the cost of network multicast compared to unicast is presented in (Chuang and Sirbu, 2001). There is a need to provide more specific study of multicast cost saving in delivering the immersive voice communication service in different network game scenarios.

In this chapter, we investigate the use of the distributed proxy architecture for providing an immersive voice communication service to MMOG. In Section 6.2, we describe the proxy location problem and formulate the problem using an Integer Lin-

ear Programming model. We also devise a greedy heuristics for the problem based on a graph algorithm. Different proxy architectures with choices of unicast and multicast implementation are also discussed. Simulation results are presented in Section 6.3. In these simulations, we compare the performance of the Integer Linear Programming model and the heuristics. We also evaluate the bandwidth cost saving of network multicast in the distributed proxy architecture based on different avatar grouping behaviours and player distribution scenarios. In addition, the effects of varying the number of proxy servers on communication delays and network bandwidth usage are investigated. Conclusions are drawn in Section 6.4.

6.2 Service Delivery Model

6.2.1 Proxy Location Problem

It is envisaged that a game service provider can have access to a multitude of servers located in different parts of the Internet. These servers are deployed as distributed proxies for providing an immersive communication service to the game. We assume that players are connected to N ISP POPs and there are P potential proxy servers. The proxy assignment is based on physical location of players so that all players connected to an ISP POP will be assigned to the same proxy. The problem is to assign N ISP POP to P proxies in a way that minimizes communication delay in the game.

The distribution of proxy servers in this architecture is based on the physical world partitioning as all players from each ISP POP are assigned to a proxy. On the other hand, in the distributed locale server architecture, the distribution of servers is based on the virtual world partitioning since all players in each locale are assigned to the same server.

A simple solution is to assign players at each ISP POP to the closest proxy. This solution does not require avatar position in the virtual world which determines communication groups in the game. However, this solution may not provide an optimal proxy assignment. We refer to this solution as a *simple heuristics*. In the next section,

we provide other algorithms that use the knowledge of communication groups in the game to minimize latency.

We assume that each proxy has enough capacity to server all players. Hence, the assignments of players to proxies are mainly based on latency optimization, not due to resource constraints. It is also assumed that processing time for creating audio scenes at proxies are negligible. The reason for this assumption is that processing times at proxies are small compared with propagation delays. Also, we focus on optimizing latency based on locations of proxies regardless of the resource constraint at each proxy, while processing times at proxies often depend on the load at these proxies.

Since we assume that each proxy is not overloaded, processing times at proxies are small and not varied. Therefore, these processing times does not have major impact on the optimization of the distributed proxy architecture which is mainly based on proxy locations.

6.2.2 Mathematical Programming Model

We use the following mathematical formulation to model the proxy location problem. Fist of all, the known parameters are defined as follows.

c_i^s: The delay cost from ISP POP i and proxy s which is defined as the delay between ISP POP i and proxy s multiplied by the number of players connected to this ISP POP.

k_{ij}^{st}: The inter-proxy delay cost occurs when proxy s is assigned to ISP POP i and proxy t is assigned to ISP POP j. This delay cost is defined as the delay between the two proxies multiplied by the number of players located at ISP POP i that are in communication with other players located at ISP POP j.

Decision variables:

$$x_i^s = \begin{cases} 1 & \text{if proxy } s \text{ is chosen for ISP POP } i \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

The objective is to minimize the total interactive communication delay of the game. The objective function is defined as follows:

Minimize:

$$\sum_{1\leq i\leq N;1\leq s\leq P} x_i^s c_i^s + \sum_{1\leq i,j\leq N;1\leq s,t\leq P;i\neq j;s\neq t} x_i^s x_j^t k_{ij}^{st} \tag{6.2}$$

subject to

$$\sum_{s=1}^{P} x_i^s = 1 \quad \forall i : 1 \leq i \leq N \tag{6.3}$$

It is interesting to see that this mathematical programming formulation is similar to the formulation for server assignment problem in distributed locale servers in Section 5.2.2. This is not surprising as the two problems have many common features. Since this formulation is non-linear we use a similar method to the previous problem in order to linearize it as follows.

Let y_{ij}^{st} be a binary decision variable that has the following property.

$$y_{ij}^{st} = \begin{cases} 1 & \text{if } x_i^s = x_j^t = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

The former problem is transformed to the following linear programming (LP).

Minimize:

$$\sum_{1\leq i\leq N;1\leq s\leq P} x_i^s c_i^s + \sum_{1\leq i,j\leq N;1\leq s,t\leq P;i\neq j;s\neq t} y_{ij}^{st} k_{ij}^{st} \tag{6.5}$$

subject to

$$\sum_{s=1}^{P} x_i^s = 1 \quad \forall i : 1 \leq i \leq N \tag{6.6}$$

$$y_{ij}^{st} \leq x_i^s \tag{6.7}$$

$$y_{ij}^{st} \leq x_j^t \tag{6.8}$$

$$x_i^s + x_j^t \leq 1 + y_{ij}^{st} \tag{6.9}$$

The constraints in equation 6.6 ensure that each ISP POP is assigned to one proxy. The constraints in equations 6.7, 6.8, and 6.9 are to ensure the condition in equation 6.4 is satisfied. Basically, these constraints ensure that if players connected to different proxies are in interactive communication, the delay cost due to inter-proxy communication is taken into account. This mathematical formulation is an Integer Linear Programming (ILP) problem.

If we consider a scenario that a game service provider only has a budget for m proxies ($m < P$), additional variables and constraints are required. Let z^s be a binary variable, the following constraints are needed:

$$z^s \geq x_i^s \tag{6.10}$$

$$\sum_{s=1}^{P} z^s \leq m \tag{6.11}$$

The constraints in equation 6.10 are to ensure that if a server is assigned to an ISP POP, this proxy is counted. The constraints in equation 6.11 are to ensure that only maximum of m proxies are used.

As there are N ISP POPs and P proxies, the number of x_i^s variables is NP. The number of y_{ij}^{st} variables depends the number of communications exchanges between players assigned to different proxies. For each ISP POP that is already assigned to a proxy, there are P different assignment possibilities of other ISP POP. As a result, the number of y_{ij}^{st} is $O(NP^2)$. Therefore, the total number of binary decision variables for this problem, which is the sum of x_i^s and y_{ij}^{st} variables, is $O(NP^2)$.

We expect that the number of binary variables for this problem can be large due to the large number of communication exchanges between players located at these POPs. For example, in an extreme case, the number of audio flow exchanges can be $N(N-1)/2$. This is different from the case of the distributed locale server architecture in which each locale only has a maximum number of eight adjacent locales. As a result, the number of y_{ij}^{st} variables in the ILP model for the distributed proxy architecture can be considerably larger than that in the distributed locale server architecture, given the same number of servers and ISP POPs/locales.

Similar to the analytical model in Section 5.2.2, this problem is also NP-hard. The proof can be carried out in a similar method as in Section 5.2.2. When the size of the problem is large, it is necessary to design a heuristics for solving the problem more efficiently. In the next section, we will provide a greedy heuristics for this problem based on a graph algorithm.

6.2.3 Heuristic Algorithms

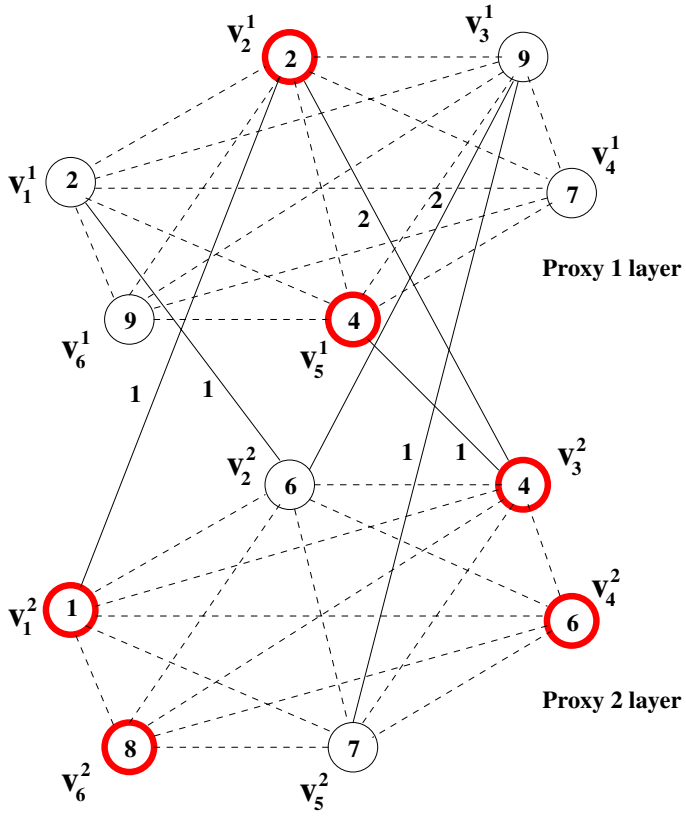
6.2.3.1 Graph Representation

We use a multi-layer graph model which is somewhat similar to the multi-layer graph in Section 5.2.3.1. The graph is $G = (V, E)$, where $|V| = PN$ and $V = \{v_i^s : 1 \leq i \leq N; 1 \leq s \leq P\}$. This graph has P layers corresponding to P proxies, and each layer has N vertices corresponding to N ISP POPs. Each vertex v_i^s (vertex i in layer s) has a delay cost c_i^s. In each layer, all vertices are connected in full-mesh by edges with zero cost. There are inter-layer edges connecting any pair of v_i^s and v_j^t, denoted as e_{ij}^{st}, given that $i \neq j$ and $s \neq t$. The cost of the edge connecting vertex i in layer s and vertex j in layer t is k_{ij}^{st}. Similar to the model in Section 5.2.3.1, the problem is to find a subgraph $G' \subseteq G$ that covers N vertices corresponding to N ISP POPs and has the minimum total vertex costs and edge costs.

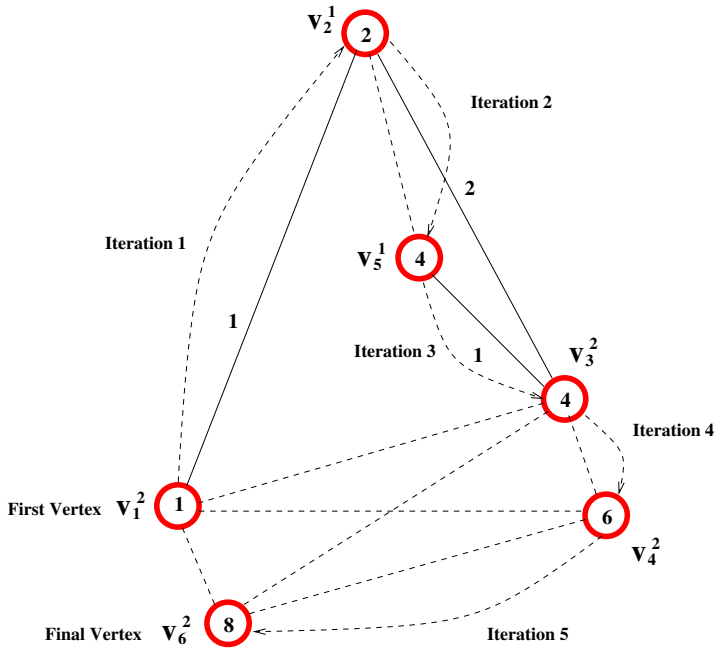
Fig. 6.1 shows a two layer graph which represents the problem of assigning 6 ISP POPs to two proxies. All vertices in each layer are connected in full-mesh by zero cost edges, denoted as broken lines. Zero cost inter-layer edges are not shown to avoid cluttering the figure. In addition, this multi-layer graph is slightly different from the previous multi-layer graph model in Section 5.2.3.1. In this model, vertices are connected in full-mesh (except there is no inter-layer edges connecting v_i^s and v_i^t). On the other hand, in the previous model, only vertices that represent adjacent locales are connected as shown in Fig. 5.1.

6.2.3.2 Greedy Heuristics

In this section, we propose a greedy heuristics which is relatively similar to the heuristic algorithm for the distributed locale server architecture in Section 5.2.3.2. The aim is to find a sub-graph $G' \subseteq G$ that covers N vertices and has the smallest sum of vertex cost and edge costs. To begin with, we choose the first vertex for which the sum of the vertex cost and its outgoing edge costs is minimum. Then, we repeat the following procedures: choose the vertex that has the minimum sum of its vertex cost and cost of edges connecting this vertex to all vertices in the existing sub-graph. When a vertex in one layer is added to the sub-graph, the corresponding vertices in



(a) Graph representation and optimal proxy assignment



(b) Construction of the sub-graph through each iteration

Figure 6.1 Graph representation of the proxy assignment problem and solution.

all other layers as well as all edges connecting to these vertices are deleted. The algorithm finishes when N vertices are covered. The pseudo code for the algorithm is presented as follows.

Pseudo code:

1. Initialize

$G'(V_{sub}, E_{sub}) = 0$;

2. Choose the first vertex v_i^s s.t ($c_i^s + \sum k_{ij}^{st}$) is minimum. Add v_i^s to G' and remove all v_i^t s.t $\{t \neq s, 1 \leq t \leq P\}$ and all edges connected to these vertices from G.

3. While $size(G') < N$ do

    4. For each $v_j^t \in G$, compute $cost_j^t = c_j^t + \sum k_{ij}^{st}$ s.t $v_i^s \in V_{sub}$.

    5. Add v_j^t to G' s.t $cost_j^t$ is minimum.

    6. Remove v_j^t from G, remove all v_j^u s.t $\{u \neq t, 1 \leq u \leq P\}$ and all edges connected to these vertices from G.

7. End

This pseudo code is slightly different from the pseudo code for the heuristic algorithm in Section 5.2.3.2. Since vertices in the layer graph are connected in full-mesh (except vertices representing the same ISP POP in different layers of the graph), there is no need to maintain a set of neighbors to G' as in the previous algorithm. In each iteration, the next vertex is selected from any vertex in G.

This algorithm has a similar run-time complexity to the heuristic algorithm in Section 5.2.3.2. If an adjacency matrix is used for the graph representation, the run-time complexity is $O(|V|^2)$, where $|V| = NP$ is the number of vertices in the multi-layer graph. If an adjacency list is used for the graph representation, the run-time complexity of this algorithm is $O((|E| + |V|)log|V|)$, where E is the number of edges in the multi-layer graph. Since this graph is almost full-mesh, the number of edges is considerably larger than the number of vertices. Hence, it is more efficient to use an adjacency matrix for the graph representation as the run-time complexity in this method does not depend on the number of edges.

The example in Fig. 6.1 shows the operation of this heuristic. The algorithm first

chooses vertex v_1^2 and removes v_1^1 and all edges connected to this vertex from G. In the first iteration, the algorithm searches the rest of the vertices in G and choose v_2^1 since it has the minimum sum of its vertex cost and cost of the edge connecting this vertex to v_1^2. v_2^2 and all edges connecting to this vertex is removed from G. In the second iteration, the algorithm searches the rest of vertices in G and chooses v_5^1 as it has the minimum sum of its vertex cost and costs of the edges connecting this vertex to v_1^2 and v_2^1. In the following iteration, v_3^2, v_4^2, and finally, v_6^2 are added to G'.

If we need to consider the maximum load of each proxy, additional constraints are required for the ILP model. For the greedy heuristics, a constraint can be put in step 4 of the algorithm. A vertex is considered only when the proxy corresponding to that vertex is not overloaded. In addition, the proxy load is incremented each time a vertex within the associated layer of that proxy is added to the sub-graph G' in step 5.

6.2.4 Different Proxy Architectures

There are two methods for forwarding audio streams among proxies: unicast and multicast. As an example, Fig. 6.2 compares the core network bandwidth usages of the peer-to-peer architecture with unicast routing (peer-to-peer), distributed proxy architectures using unicast routing (proxy unicast) and multicast routing (proxy multicast). In the figure, c_1, c_2 and c_3 denote the bandwidth costs of links. The figure shows the network bandwidth requirements of different architectures for delivering the audio stream from avatar a to all other avatars. As shown from the figure, the peer-to-peer architecture uses most resources, proxy unicast uses less, and proxy multicast uses the least. The quantitative comparison of network bandwidth usages of these architectures in different game delivery scenarios will be provided in the simulation study in the next section.

There are two approaches for implementing multicast for group communication: IP multicast and application layer multicast. In IP multicast (Deering, 1989), multicast functions are implemented at IP routers. The router makes a copy of IP packet and forward it to other links as indicated by a multicast tree. However, IP multicast has not been commercially implemented widely in the Internet due to the lack of mul-

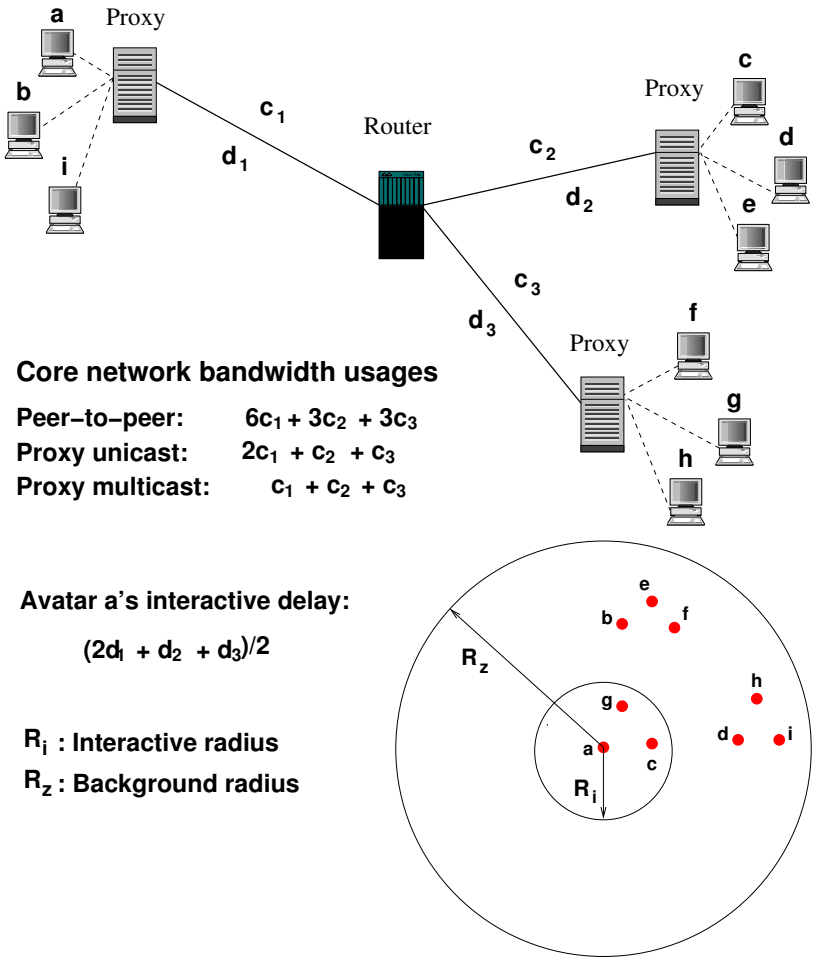


Figure 6.2 Network bandwidth usages and interactive delay calculation associated with avatar "a".

ticast support in routers in various Internet domains. In application layer multicast, all multicast functionality including membership management and packet replication are implemented at end systems instead of routers. These architectures are currently deployed for media streaming and transmission of content from an original server to replicated servers in a CDN. Application layer multicast does not require changes to the underlying network, thus, the deployment is faster compared to IP multicast. In addition, application layer multicast maintain stateless nature of the network while IP multicast needs to maintain per group state information at IP routers.

In this thesis, we consider IP multicast which is the most efficient and effective approach. As IP multicast also has better bandwidth saving than application layer multicast, the network bandwidth study of IP multicast can be used as a benchmark for application layer multicast.

6.3 Simulation Experiments

In this simulation study, the following issues are investigated:

- Comparison of different proxy assignment algorithms.
- Efficiency of network multicast.
- Impact of varying the number of proxies.
- Comparison of network bandwidth usages in different architectures.

6.3.1 Simulation Setup

We use GT-ITM topology generator to model the Internet topology. A transit-stub graph of 5000 nodes, with an average node degree of 3.9, is generated for the simulations. This topology is larger than the 600 node topology used in chapters 4 and 5 due to the number of proxies in this simulations can be large (from 50 to 100). Details of this topology is provided in Appendix B.

In this topology, ISP POPs are randomly chosen among the set of nodes, and the number of game players located at these POPs are randomly generated based on a uniform distribution. The topology generator parameters are chosen such that the maximum propagation delay in the shortest path between two nodes is 300ms. Unless otherwise stated, the numbers of POPs and game players are 100 and 5000, respectively, and the physical/virtual world correlation is 0. The number of crowds/clans in a crowd/clan based virtual world is 50 and 250, respectively.

We simulate two proxy distribution scenarios and also vary the number of proxies and ISP POPs in each scenario. In the first scenario, proxies are deployed at random locations in the network. In the second scenario, we envisage that proxies are often placed at the edge of the network close to ISP POP. Hence, in the simulation, a number of proxies are randomly chosen from the set of ISP POPs. If a service provider has a budget for a large number of proxies, it is possible to deploy a proxy at each ISP POP. In this case, the distributed proxy architecture provides the lowest interactive communication delay.

We compare the performance of the distributed proxy architecture with the central server architecture and the peer-to-peer architecture. In the central server architecture, an optimal central server is chosen among a set of potential servers with the objective to minimize the average delay from all game players to that server as described in Chapter 4.

6.3.2 Investigation of Proxy Assignment Algorithms

In this section, we compare the performance of three proxy assignment algorithms proposed in the previous section. These include the integer linear programming model for optimal server assignment, the greedy heuristics, and the simple heuristics. For each algorithm, we calculate the total communication delay cost as defined in Section 6.2.2. Cplex optimization software [1] is used for solving the ILP model. The greedy heuristics is implemented in C++. We measure the running times of the ILP model and the greedy heuristics on a Pentium IV 2Ghz Linux server with 1.5GB of RAM.

[1] Cplex optimization software, available at http://www.ilog.com

Proxies	Variables	Optimal	Time	Greedy	Time	Gap(%)	Simple	Gap(%)
3	1125	111784	0.08(s)	116030	<0.01(s)	3.7	120333	7.7
5	3625	80003	5.6(s)	80477	<0.01(s)	0.6	87586	9.5
7	7525	97989	5min24s	103757	<0.01(s)	5.8	110449	13
10	16000	76618	30min12s	80346	<0.01(s)	4.8	81704	6.6

Table 6.1
Comparison between the optimal proxy assignment, the greedy heuristics, and the simple heuristics with respect to total interactive communication delay.

In the first experiment, we simulate a simple scenario in which the number of ISP POPs is 25 and the number of potential proxy servers is 10, and the number of players is 1000. These proxies are randomly chosen among network nodes. Table 6.1 shows the total interactive communication delay (in ms) when using the proposed proxy assignment algorithms. The gaps show the differences (in %) between each of these heuristics and the optimal results. It is shown that the greedy heuristics performs considerably better than the simple heuristics. The ILP model provides the lowest delay but is not scalable. Specifically, it takes more than 30 minutes for Cplex to solve a simple scenario which consists of only 25 POPs and 10 proxies. In this case, the running time for the greedy heuristics is less than 10ms.

Fig. 6.3 shows the actual assignments of each algorithm when there are three potential proxies. The position of 25 ISP POPs and three proxies are shown although the network topology is not shown to avoid cluttering the figure. It is shown that there are interesting differences in the way each algorithm assigns ISP POPs to proxies. The ILP model may assign ISP POPs to a farther proxy with the objective to minimize the overall communication delay. On the other hand, the simple heuristics strictly assigns ISP POP to the closest proxy, regardless of the delay between these proxies and audio flow exchanges between them. The greedy heuristics provides a solution that lies between the optimal result and the simple heuristics.

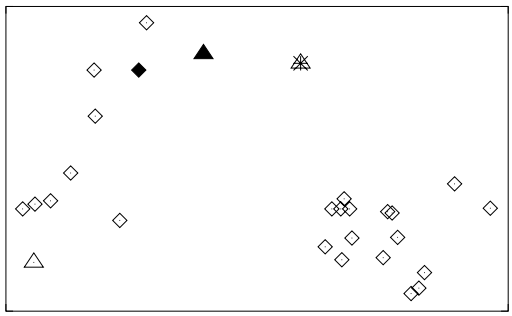
It is also noted that the ILP model only uses two proxies out of three available proxies. It is due to the fact that the ILP model ensures optimal delay performance of the distributed proxy architecture, regardless of the number of proxy servers actually used. As it will be shown later, when using simple heuristics for proxy assignment, the performance of the distributed proxy architecture may be worse than the opti-

mal central server when only a small number of proxies are available. In the worst case, due to unsuitable position of potential proxies with respect to ISP POPs, the ILP model may choose only one proxy server and this architecture become a central server architecture.

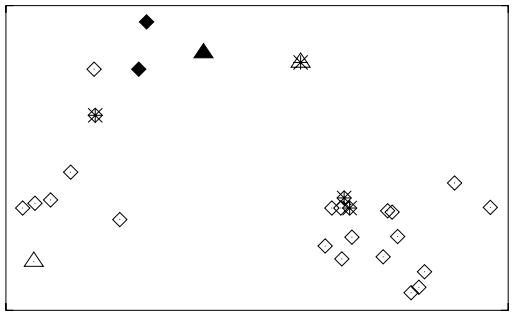
In the simple heuristics assignment in Fig. 6.3c, it is interesting to see that physical distance does not always correlate to delay in routing path. An ISP POP that is physically closer to one proxy may be assigned to another proxy which is closer in terms of routing path latency. This is due to the network topology that connects these nodes.

In the second experiment, we increase the number of POPs to 100 and the number of potential proxies to 25. Proxies are randomly chosen from network nodes. Table 6.2 compares the performance of the ILP model and the greedy heuristics. The table shows that the ILP is very unscalable as the running time of this algorithm increases from about 3 seconds to more than 58 hours when the number of proxies increases from 3 to 5. Table 6.3 shows the superior performance of the greedy heuristics compared with the simple heuristics. The gap between these algorithms ranges from 11% to above 31%. In addition, the greedy heuristics has very low running time and is very scalable.

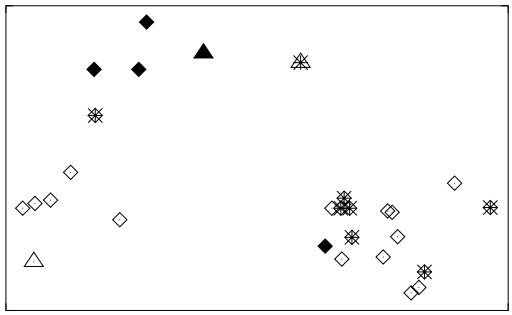
In the third experiment, the number of potential proxies is increased to 50 and these proxies are located at ISP POPs. Table 6.4 shows the results when there are 100 POPs while Table 6.5 shows the results when there are 50 POPs. As shown from the table, the greedy heuristics still shows superior performance compared with the simple heuristics. However, when the number of available proxies is large, the gap between the greedy heuristics and the simple heuristics is smaller. Especially, when the number of proxies gets close to the number of ISP POPs, the simple heuristics becomes more effective, as expected, and gets close to the performance of the greedy heuristics. Finally, it is further demonstrated that the running time of the greedy heuristics is still low even when the number of proxies increases up to 50.



(a) Optimal



(b) Greedy Heuristics



(c) Simple Heuristics

Figure 6.3 Comparison of proxy assignment algorithms.

Number of proxies	Variables	Optimal	Time	Greedy	Time	Gap(%)
2	4434	500324	0.3(s)	507919	0.01(s)	1.5
3	13200	480611	3.37(s)	529323	0.02(s)	10
5	42840	374020	58h20min	426650	0.06(s)	14

Table 6.2
Comparison between optimal results and greedy heuristic results in the total interactive communication delay.

Number of proxies	Greedy	Time	Simple	Gap(%)
2	507919	0.01(s)	565921	11.4
3	529323	0.02(s)	630886	19.2
5	426650	0.06(s)	561545	31.6
10	350412	0.24(s)	391674	11.8
15	408794	0.51(s)	474542	16.1
20	393401	0.87(s)	436987	11.1
25	374864	1.3(s)	419541	11.9

Table 6.3
Comparison between the greedy heuristics and the simple heuristics in the total interactive communication delay.

6.3.3 Investigation with Proxy Architectures for Different Player Aggregation Behaviours

In this section, we investigate the bandwidth saving of multicast when varying virtual/physical world correlation in crowd/clan based games. In the following simulations, a proxy is located at each ISP POP. We calculate the network bandwidth usage of the peer-to-peer architecture and the proxy multicast architecture.

Fig. 6.4 shows the network bandwidth requirements of the peer-to-peer architecture and the proxy multicast architecture when the correlation increases from 0 to 1. As the correlation increases, there are fewer audio flow exchanges between ISP POPs and the network bandwidth usages of both architectures decrease considerably. In particular, in crowd based games, the bandwidth usage ratio of peer-to-peer over an optimal central server decreases from above 20 at no correlation to about 6 at a correlation of 1. The bandwidth usage ratio of proxy multicast over an optimal central server also decreases from 8 at no correlation to 1 at a correlation of 1. In addition, when using proxy multicast, the network bandwidth usage is reduced by

Number of proxies	Greedy	Time	Simple	Gap(%)
2	465818	0.02(s)	495067	6.2
3	513879	0.03(s)	575849	12.1
10	553966	0.35(s)	641702	15.9
20	459463	1.41(s)	533199	16
30	454349	2.12(s)	522272	15
40	418743	3.7(s)	444598	6.2
50	424455	6.87(s)	441280	4

Table 6.4
Comparison between the greedy heuristic and the simple heuristic in the total interactive communication delay.

Number of proxies	Greedy	Time	Simple	Gap(%)
2	440946	<0.01(s)	544444	23.5
5	458035	0.01(s)	533251	16.4
10	466955	0.06(s)	481700	3.2
20	369131	0.21(s)	397349	7.7
30	357900	0.47(s)	376559	5.2
40	346293	0.82(s)	361841	4.5
50	343464	1.37(s)	343498	0

Table 6.5
Comparison between the greedy heuristics and the simple heuristics in the total interactive communication delay.

more than 50%, at low correlation, and by about 75%, at high correlation, compared with the peer-to-peer architecture. The bandwidth requirements in clan-based games are smaller (due to smaller numbers of avatars within hearing range) and follow a similar characteristic to crowd-based games.

In the second experiment, we study the effect of varying avatar density on the network bandwidth requirements of peer-to-peer, proxy unicast and proxy multicast. The results show that the avatar density does not affect the network bandwidth usage of a central server. As shown in Fig. 6.5a, in a loner based virtual world, the bandwidth usages of peer-to-peer and proxy unicast are similar. When the average number of avatars within hearing ranges is varied from 0.5 to 5, the ratio of bandwidth usage of peer-to-peer/proxy unicast over the central server architecture increases linearly from 0.2 to about 2. Proxy multicast bandwidth requirement increases almost lin-

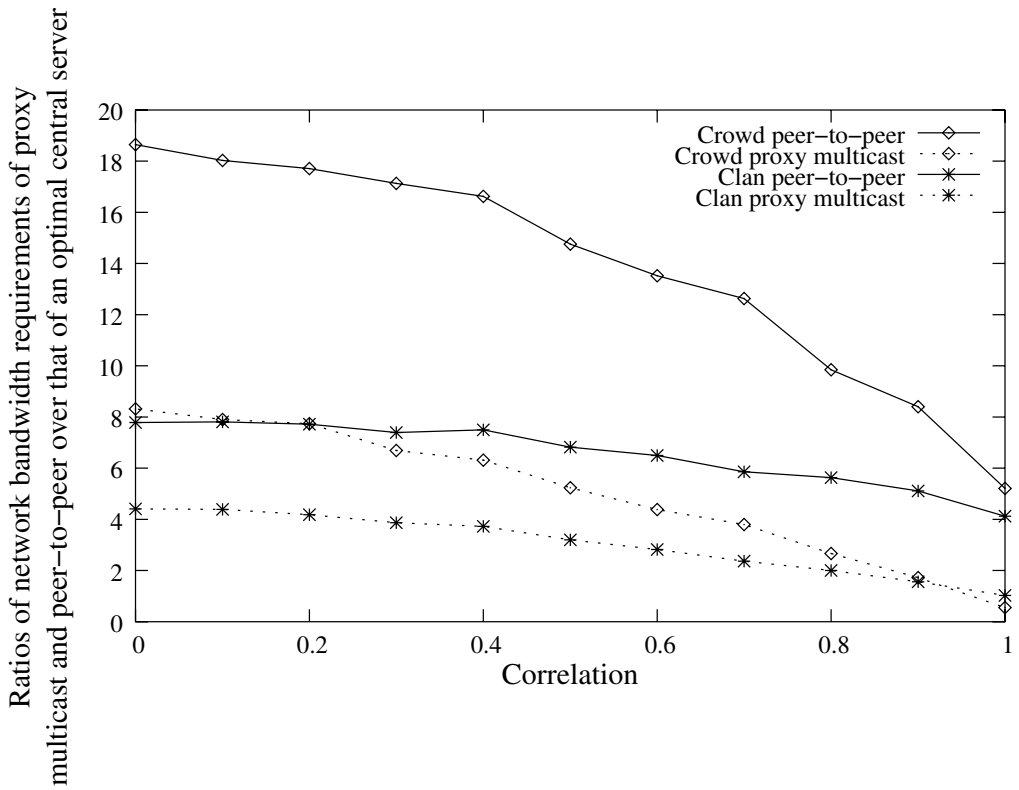
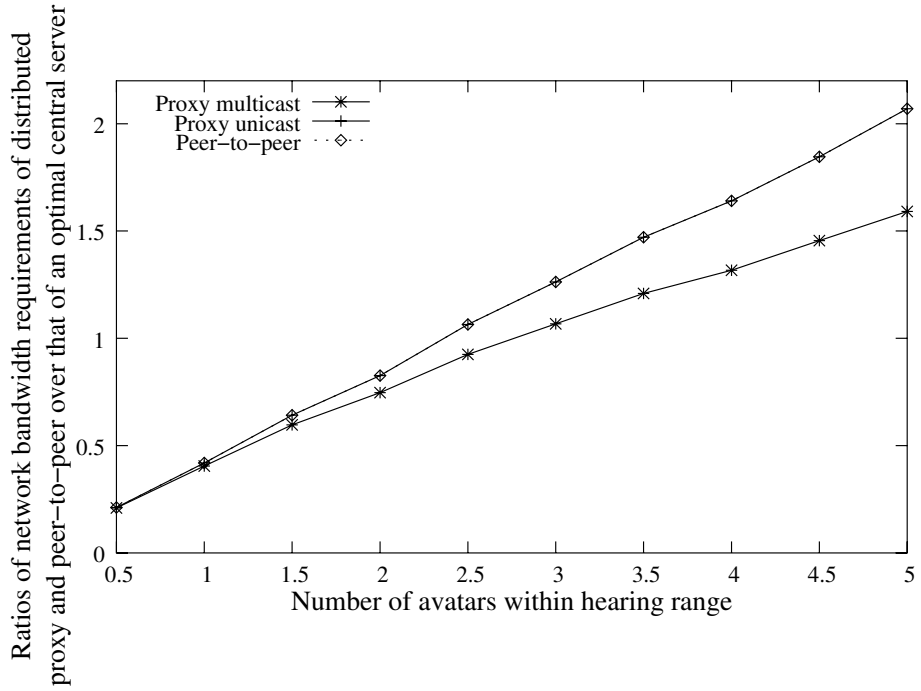


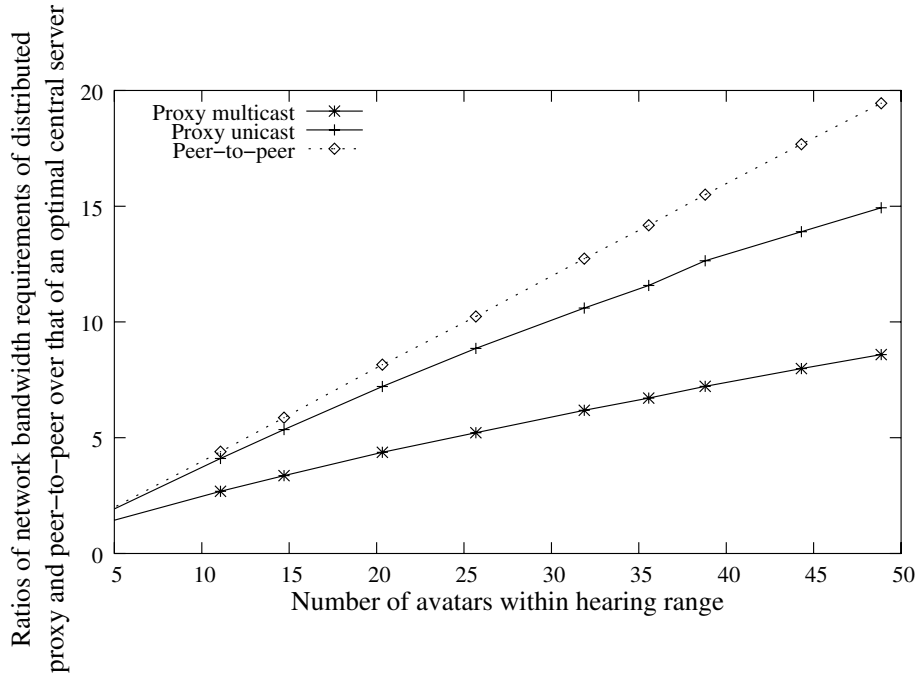
Figure 6.4 Effect of varying virtual/physical world correlation on network bandwidth requirements in crowd/clan based games

early, and is only slightly lower than that of peer-to-peer. The bandwidth saving of multicast here is not significant due to a large number of small multicast groups.

On the other hand, the bandwidth saving of multicast is more significant in a clan/crowd based game due to a larger number of avatars in each multicast group. As indicated in Fig. 6.5b, when the average number of avatars within hearing range in a clan/crowd based virtual world is varied from 5 to 50, the network bandwidth requirement ratio of peer-to-peer versus central server increases linearly from 2 to nearly 20. However, the bandwidth requirement of proxy multicast is significantly lower, especially, at high avatar densities. Specifically, at high densities, the network bandwidth usages of proxy unicast and proxy multicast are reduced by 25% and more than 50%,



(a) Loners



(b) Clans and Crowds

Figure 6.5 Effect of varying density on network bandwidth requirements in loner and crowd based games

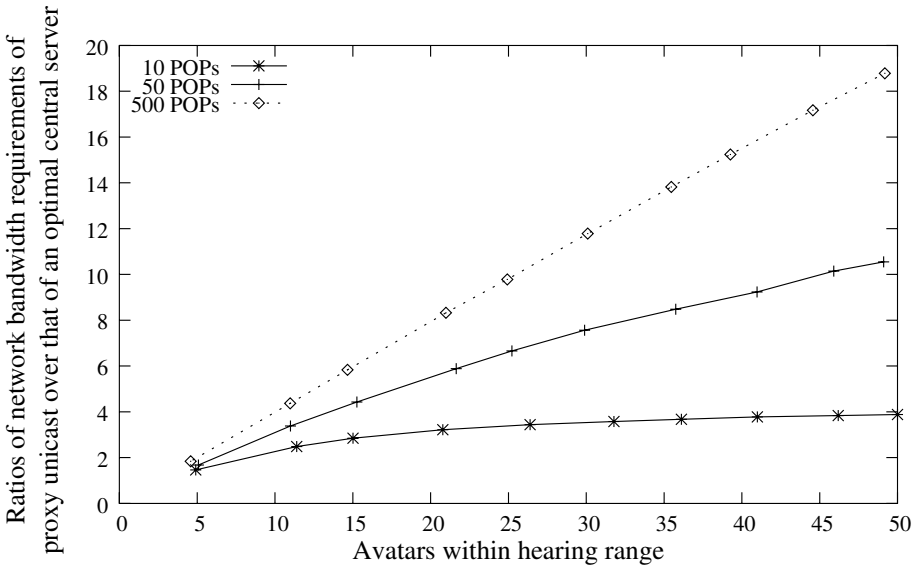
respectively, compared with the peer-to-peer architecture.

6.3.4 Efficiency of Multicast

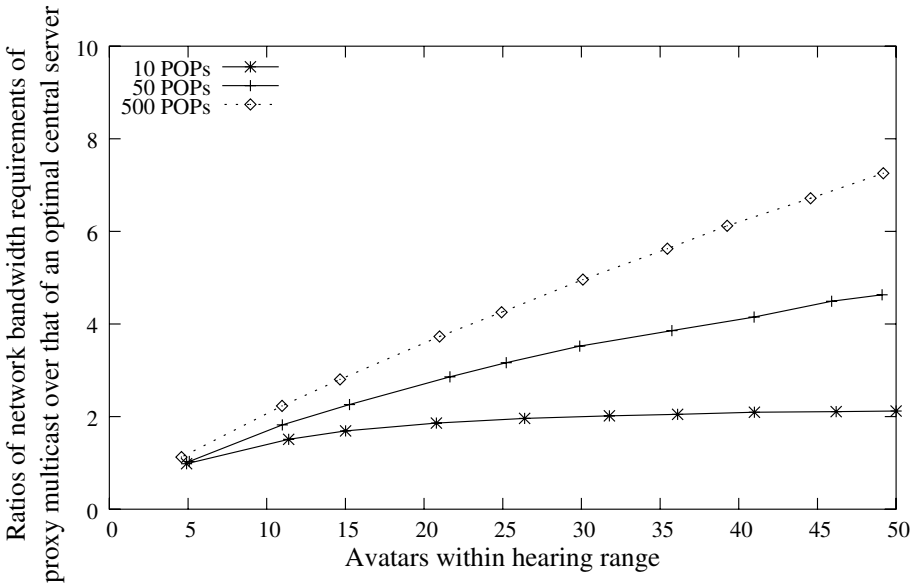
The results in the previous section show that multicast is more suitable for games with crowds and large clans, and not effective for loner based games. In addition, when the number of avatars in hearing range is large, it is more likely that some of avatars are connected to the same proxy, therefore, the bandwidth usage of proxy unicast is lower than that of peer-to-peer.

We investigate the effect of game players distribution on bandwidth requirements of proxy unicast and proxy multicast architectures by varying the number of POPs that game players are connected to in a crowd based game. We assume that there is a proxy deployed at each POP and the correlation is set to 0. The results indicate that the network bandwidth usage of the peer-to-peer architecture does not depend on the number of POPs since a player needs to send separate audio streams to listeners who resides at the same POPs. However, a proxy only needs to send one streams for listeners connected to another proxy. Hence, assuming that a proxy is located at each ISP POP, the number of ISP POPs has a considerable effect on the network bandwidth usage of the distributed proxy architecture.

As indicated in Fig. 6.6b, the bandwidth saving of proxy multicast less significant when game players are connected to a large number of POPs. When game players are connected to a small number of POPs, there are more overlaps in each multicast group, therefore, the advantage of multicast is more significant. Specifically, the bandwidth saving of multicast improves by three times when the number of POPs decreases from 500 to 10. In addition, as shown in Fig. 6.6a, the network bandwidth usage of proxy unicast also reduces by five times when the number of POPs reduces from 500 to 10. However, in most cases, the network bandwidth usage of proxy unicast architecture is about twice that of proxy multicast.



(a) Proxy unicast



(b) Proxy multicast

Figure 6.6 Effect of varying the number of POPs on network bandwidth requirements of proxy multicast and unicast

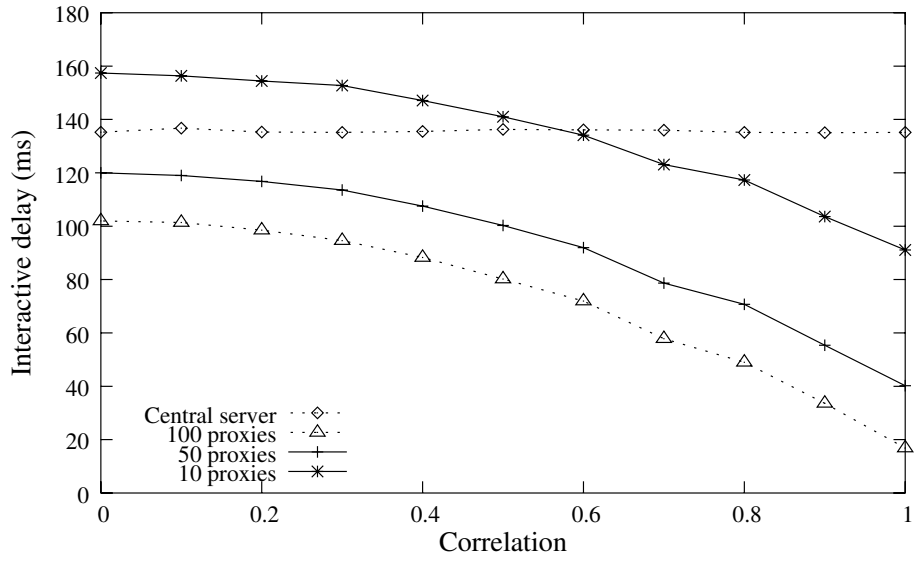
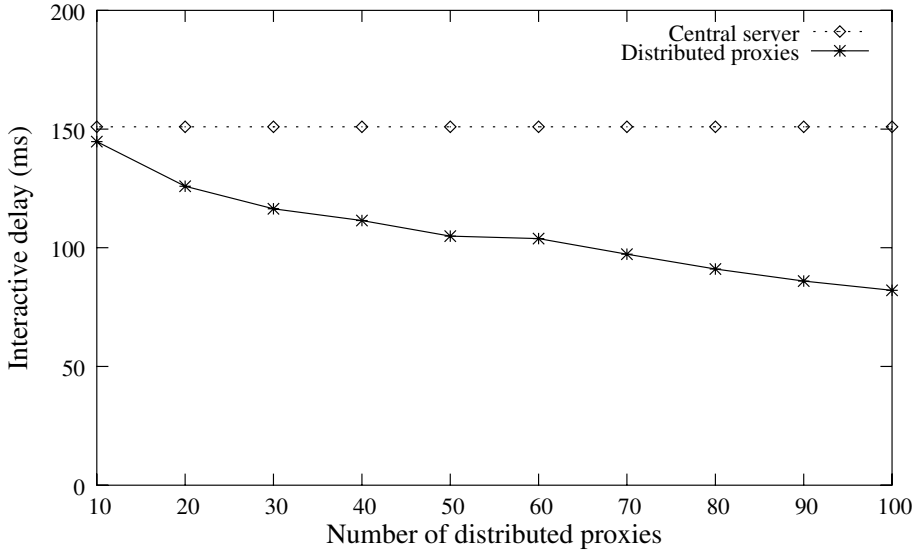


Figure 6.7 Effect of varying virtual/physical world correlation on interactive delays

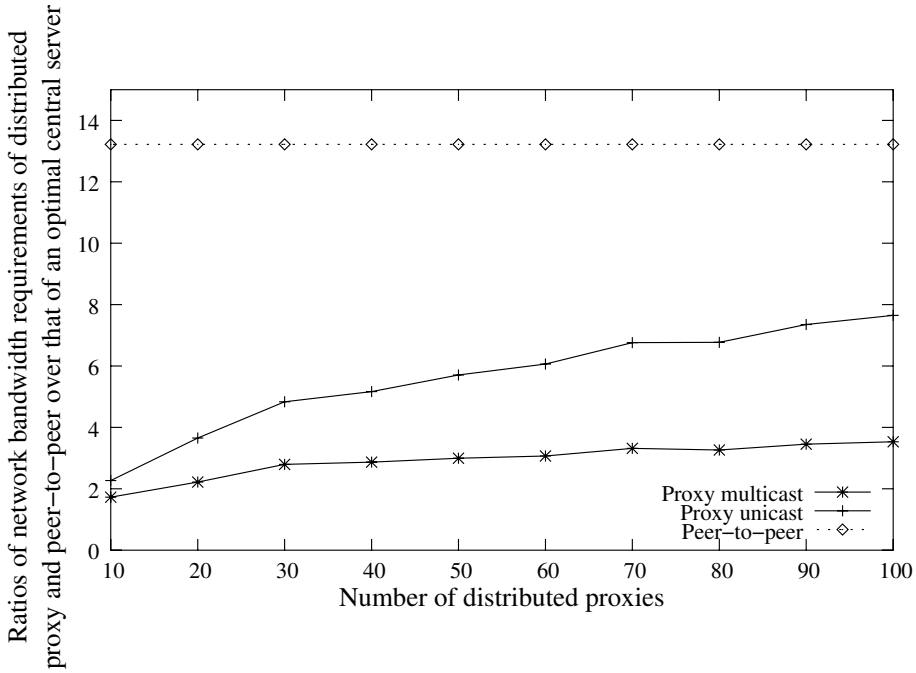
6.3.5 Effect of Varying Number of POPs and Servers

In this section, we investigate the effect of varying the number of proxy servers for a range of correlation parameters. The number of proxies is chosen as 10, 50, and 100 (the latter is equal to the number of POPs). These proxies are located at ISP POPs and simple heuristics is used for proxy assignment. As shown in Fig. 6.7, while the delay of the central server does not depend on the correlation, the delay of the proxy architecture reduces when the correlation increases. When a proxy is located at every POP, the distributed proxy architecture has the smallest delay due to direct routing paths between POPs. When the number of proxies is half of the number of POPs (50 proxies), the interactive delay of the proxy architecture increases but is still significantly lower than that of the central server, especially, at high correlation. However, when the number of proxies is one tenth the number of POPs (10 proxies), this delay increases significantly and is even higher than the interactive delay of the central server when the correlation is under 0.5.

Fig. 6.8 investigates the interactive delay and network bandwidth requirements of the distributed proxy architecture in a crowd based game when the number of prox-



(a) Interactive delay



(b) Network bandwidth requirements

Figure 6.8 Effect of varying number of proxies on interactive delay and network bandwidth requirements of distributed proxy architectures

ies is varied from 10 to 100 and the physical/virtual world correlation is 0.5. As indicated in Fig. 6.8a, when the number of proxies is reduced, the interactive delay increases significantly from just above half of the delay of the central server to nearly one. This is due to the fact that when the number of proxy servers is reduced, the distances from proxies to some POPs are increased. As a result, longer routing paths are required between two POPs when audio flows are routed through these proxies. On the other hand, as shown in Fig. 6.8b, when the number of proxies is reduced from 100 to 10, the network bandwidth requirements of proxy unicast and proxy multicast decrease linearly by about 60% and 50%, respectively. In this case, since only one audio stream is required from a POP to a proxy for each player's voice transmission, reducing the number of proxies will reduce the bandwidth usages of proxy unicast and proxy multicast.

6.4 Conclusions

In this chapter, several proxy assignment algorithms are proposed for the distributed proxy architecture. We first formulate the proxy assignment problem as an Integer Linear Program. This formulation provides optimal proxy assignment but has a very high computation complexity. Hence, we adapt the multi-layer graph representation proposed in Chapter 5 and devise a greedy heuristics based on this graph. The simulation results show that the greedy heuristics has good delay performance and low run time complexity. This heuristics also performs considerably better than a simple heuristics which assigns each ISP POP to the closest proxy.

In addition, a quantitative simulation study has been carried out to evaluate the bandwidth saving of network multicast in the distributed proxy architectures in different avatar grouping behaviours and player distribution scenarios. The effect of varying the number of proxy servers on communication delays and network bandwidth usages are also investigated.

Multicast is effective only when the average number of avatars in hearing ranges is large (i.e crowds). In these scenarios, when game players are connected to a small number of ISP POPs, the bandwidth cost saving of multicast is significant. How-

ever, when game players are widely spread, the efficiency of multicast is reduced significantly. The proxy unicast architecture also has smaller bandwidth usage than the peer-to-peer architecture, especially when game players are connected to a small number of POPs. This is due to the fact that peer-to-peer can send multiple copies of the same stream to other players connected to the same POPs. Although this bandwidth usage reduction is not as significant as multicast, the proxy unicast is worth being considered since the implementation of network multicast over large-scale infrastructure is a costly task.

The number of proxies has a significant impact on the latency. If only a small number of proxies are available (e.g., less than one tenth of the number of POPs) distributed proxies can have higher delay than the central server when using the simple heuristics for proxy assignment. However, if a large number of proxy servers are available over the Internet, the distributed proxy architecture would be a cost effective delivery architecture since it reduces the delay of the central server architecture and solves access bandwidth problems of the peer-to-peer architecture. This scenario is possible since ISPs are generally willing to put game servers (or proxy servers) in their networks in order to attract more game players (or Internet service subscribers), and increase their revenues.

Chapter 7

Comparison of Architectures

7.1 Introduction

In Chapter 3, we introduced two distributed server architectures for the creation of immersive audio scenes. The first architecture is distributed locale servers, which is based on partitioning the virtual world to smaller areas called locales and assigning each locale to a server. In the second architecture, a group of participants in a given geographical location are assigned to a nearby server. This server, which is referred to as a proxy, is responsible for the creation of audio scenes for its attached participants. This architecture is referred to as the distributed proxy architecture. In essence, these two architectures are 'dual' of each other; one based on partitioning of virtual world (into locales) and the other based on the partitioning of physical world (into geographical regions) and then assigning a suitable server to each partition.

The main focus of this chapter is to provide a quantitative comparison of the performance of these two distributed architectures in different game delivery scenarios. A simulation study is carried out to address the following key questions:

- With the same number of servers, which architecture will provide better delay performance.
- What is the effect of increasing the number of servers on the performance of each architecture.

- What are the advantages and disadvantages of these architectures under various avatar aggregation behaviours.

In addition, in previous chapters, comparative results between each of these two architectures with the central server and the peer-to-peer architecture have been provided. In this chapter, we present a summary of the performance evaluation of different deliver architectures and recommendations on the effective deployment of these architectures in different scenarios.

The rest of this chapter is organized as follows: In Section 7.2, we compare requirements of different delivery architectures. Performance evaluation of these architectures is presented in Section 7.3. Finally, in Section 7.4, we provide a summary of overall performance evaluations and recommendations of delivery architectures in different game scenarios. In particular, these recommendations are given based on the server resource availability, game's avatar aggregation behaviors, physical/virtual world correlation and multicast.

7.2 Comparisons of Architectural Requirements

7.2.1 Server Assignment Algorithms

It is assumed that a game service provider can have access to a number of servers located over the Internet. The provider could choose a server site based on distribution of participants. Since latency is very important in interactive communication, players need to be assigned to servers efficiently in order to minimize communication delays.

In Chapter 4, we introduced two simple optimization objectives for choosing the optimal central server. The server selected is the nearest possible server site to the 'centre of mass' (in terms of network delay) of participants whose avatars are not isolated. In distributed server architectures, server assignment is a more complex problem.

In Chapter 5, we produced an integer linear programming (ILP) formulation and a greedy heuristic for optimal server assignment in the distributed locale server archi-

tecture. Our optimization algorithms assign servers to locales in a way that minimizes total communication delay perceived by avatars. In Chapter 6, we devised an ILP, a greedy heuristics, and a simple heuristics for server assignment in the distributed proxy architecture. Since these two architectures are dual of each other, there are some similarities in server assignment algorithms of these architectures.

It is interesting to see that the ILP model in each of these two distributed server architectures has the same mathematical characteristics. The differences are only in the ways decision variables represent in each architecture. For example, in the distributed locale server architecture, x_i^s denote whether server s is chosen for locale i while these variables denote whether proxy s is chosen for ISP POP i in the distributed proxy architecture. In the distributed locale server architecture, y_{ij}^{st} denote communication between players in adjacent locales while in the distributed proxy architecture, they denotes communication exchanges between players assigned to different proxies.

However, as discussed in Section 6.2.2, with the same number of locale/ISP POPs and servers/proxies, the ILP model for the distributed proxy architecture often has larger number of variables, hence, is more difficult to solve. This is due to a larger number of interaction between players located in different ISP POPs compared to that between players in adjacent locales. Simulation results in Section 5.3 and Section 6.3 also shows that the ILP model for the distributed locale server architecture has a significantly lower run time. However, for large problems, heuristics are required for both architectures.

The heuristics for both of these distributed server architectures are based on a multi-layer graph algorithms. The results show that the greedy heuristics for the distributed proxy architecture runs faster than the heuristics for the distributed locale servers since it does not need to maintain a set of neighbor nodes as discussed in Section 6.2.3. However, the gap between this algorithm and optimal result is larger than that between the greedy heuristics and optimal result in the distributed locale server architecture. In addition, a common approach of deploying proxy servers is to assign participants to the closest proxy. We refer to this as a simple heuristics. This approach does not require any knowledge of avatar distribution in the virtual world.

7.2.2 Impact of Avatar Movements and Player Distribution

Avatar movements have no impact on the central server architecture. In the distributed locale server architecture, movements of avatars in the virtual world change the composition of locales. This means that as time goes by, the current partition and assignments of locales to servers drifts from optimal. In this situation, re-assignment of locales to servers is necessary to reduce the communication delay.

While avatar movements may occur constantly, joining and leaving the game by participants tend to occur on a longer time scale. It is known that players tend to join the game in late afternoon and evening in the geographic areas where the players come from (Armitage, 2001). Especially, the research in (Feng and Feng, 2003) shows that game players in different geographical regions tend to join and leave the game in few different time blocks during a day due to time zone differences between these regions. In addition to avatar movements, this behavior may also affect optimal server allocation.

Avatar movements have less impact on distributed proxies since each player is assigned to the same proxy all the time. However, when each player moves and comes into hearing ranges of new players, the audio stream from that player must be routed to the proxies associated with all these players. In addition, that player also has to receive all audio streams from these players. As a result, the distributed proxy architecture requires very dynamic interactions between proxy servers for sending/receiving audio flows.

Changes in the distribution of game participants in diverse geographical regions due to time zone differences may affect the location of the optimal central server as discussed in Chapter 4. In distributed locale servers, these changes may lead to the need to reevaluate optimal assignment of locales to servers. In distributed proxies, newly-joined participants are just connected to the closest proxies, at least based on the simple heuristics. This does not affect the current server assignment.

7.2.3 Server Resource Requirements

There are two key resources that should be considered at each server site: *server network bandwidth* and *server processing capacity*. Server network bandwidth is the bandwidth of the connection from the server to the Internet. For example, if a server is located in a high speed LAN, the server network bandwidth is limited by the bandwidth connection from that LAN to the Internet. The server processing capacity sets a limit on the number of participants the server can support. Computing an audio scene for each participant requires certain amount of computation resource and the total computation resource requirement is proportional to the number of players. The server network bandwidth requirement is also proportional to the number of players connected to the server.

To provide the immersive voice service for MMOG, the central server architecture requires very high processing power and server network bandwidth. The distribution of servers in distributed server architectures does not only enhance the processing scalability but also reduces the server network bandwidth requirements compared to the central server architecture.

7.3 Simulation Experiments

7.3.1 Simulation Setup

The GT-ITM topology generator is used to model the Internet topology. In particular, we use a transit-stub graph of 5000 nodes, comprising of five transit domains, with the average node degree of 3.9, representing five different geographic regions. Each transit domain has on average ten transit nodes, each transit node connects to nine stub Autonomous Systems (ASs), representing the connectivity of different ASs in each region. We randomly place two ISP POPs at nine stub ASs belonging to each transit nodes, which results in 100 ISP POPs widely spread on the network. The number of game players connected to each POP is randomly generated based on a uniform distribution with an average of 50. Potential servers are randomly placed in these regions. The topology generator parameters are chosen such that the the

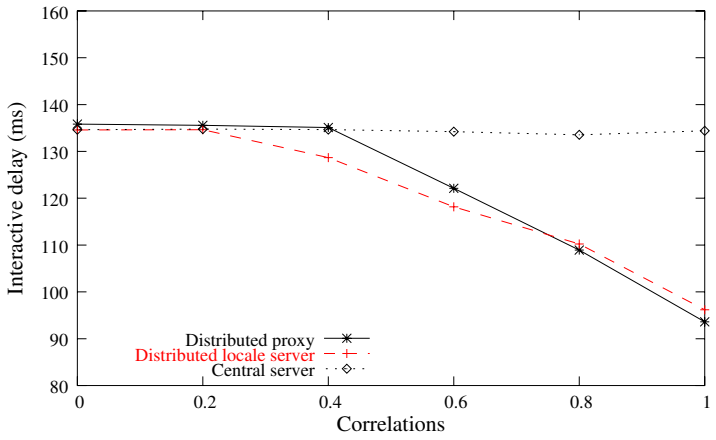
maximum propagation delay in the shortest path between two nodes is 300ms. Unless otherwise stated, the following parameters are used for simulations. The average number of avatars in an avatar's interactive zone in crowd/clan based virtual world is 2 and the physical/virtual world correlation is zero. The number of crowds and clans is 50 and 250, respectively. Details of simulation conditions are provided in Appendix B.

In each architecture, we aim to obtain an optimal server assignment or close solution to an optimal result. In the distributed locale server architecture, Cplex optimization software [1] is used to solve the ILP problem in order to achieve optimal server assignments. In distributed proxy architecture, the greedy heuristics is used for proxy assignment since it is not possible to solve by the ILP model. The simple heuristics is also used in some scenarios where proxy servers are deployed at ISP POPs. In the central server architecture, an optimal central server is chosen among a set of potential servers.

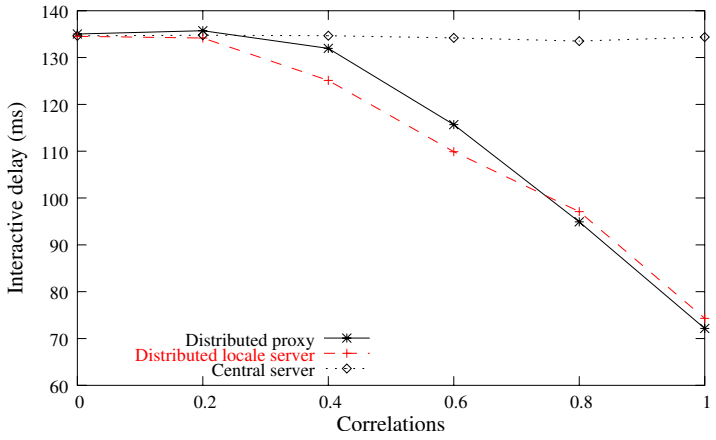
7.3.2 Interactive Delay

In the first experiment, we randomly select 50 potential servers from all of the 5000 network nodes. We simulate three scenarios in which the number of servers in each scenario is set to 5, 20, and 50. In each scenario, we also vary the physical/virtual world correlation parameters. An optimal central server is chosen from the set of selected distributed servers. Fig. 7.1 shows the delay performance of the distributed locale server architecture and the distributed proxy architecture in each simulation scenario. As shown from the figure, these two architectures result in relatively similar delay performance. At no correlation, the interactive delay of these architecture is similar to that of the central server architecture. This delay reduces considerably when either the correlation increases or the number of servers is increased. In particular, when 50 servers are deployed, the interactive delay of the distributed proxy architecture is reduced by more than 50% at a correlation of 1. The distributed locale server architecture has slightly lower delay when the number of servers is 5 and 20, and the correlation is in a range from 0.2 to 0.7. However, when 50 servers are

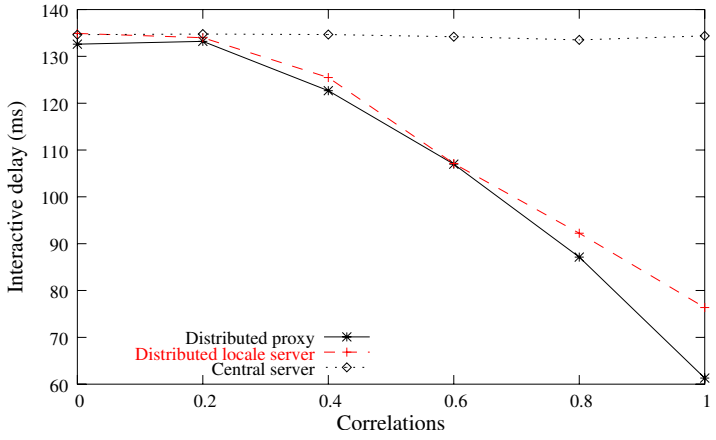
[1] Cplex optimization software, available at http://www.ilog.com



(a) 5 servers

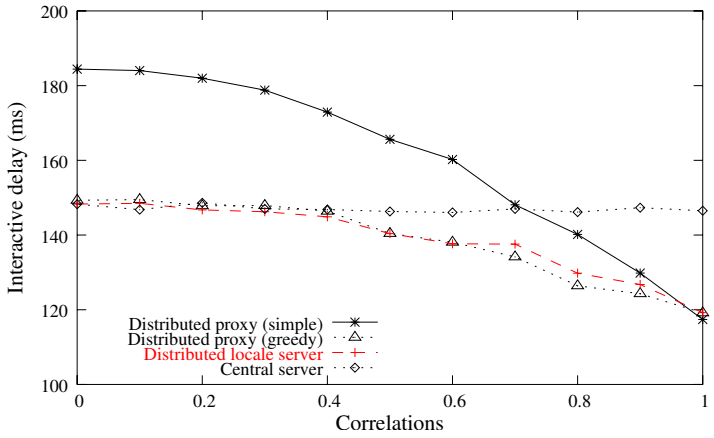


(b) 20 servers

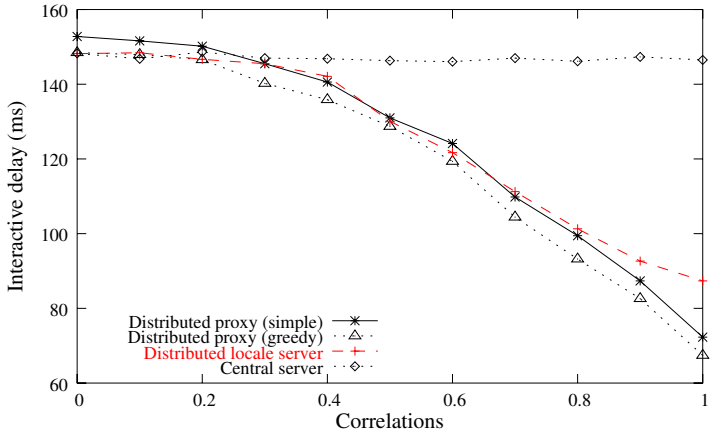


(c) 50 servers

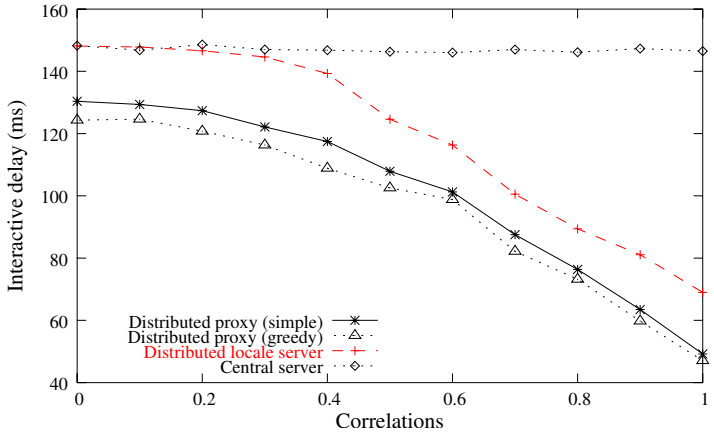
Figure 7.1 Comparison in interactive delays between distributed proxies and distributed locale servers when varying the physical/virtual world correlation.



(a) 5 servers

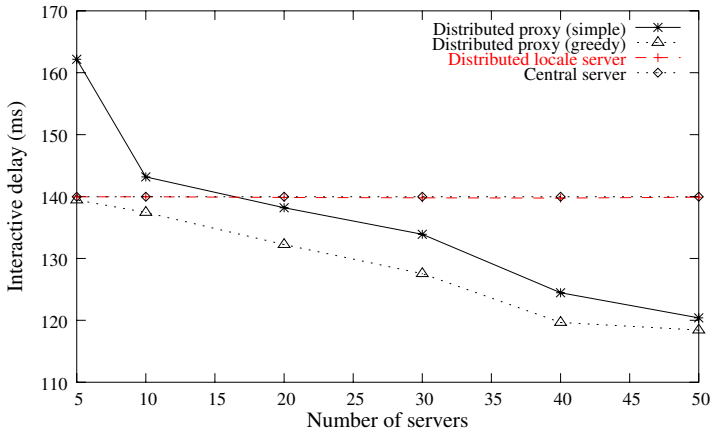


(b) 20 servers

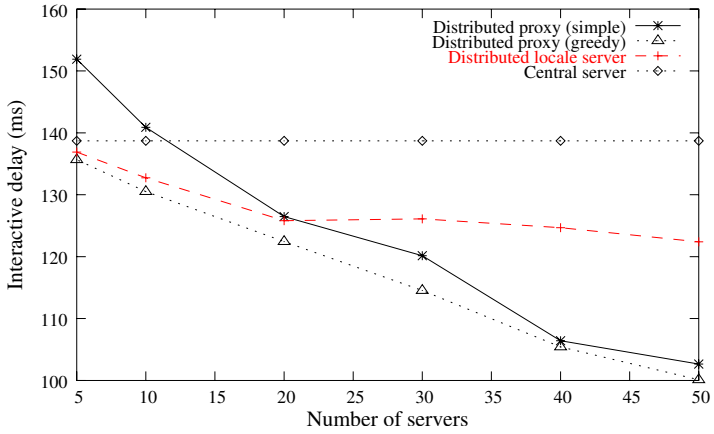


(c) 50 servers

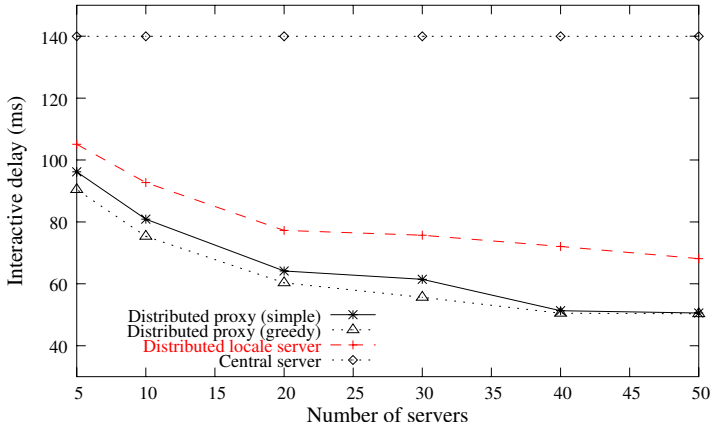
Figure 7.2 Comparison in interactive delays between distributed proxies and distributed locale servers when varying the physical/virtual world correlation.



(a) Physical/virtual world correlation of 0



(b) Physical/virtual world correlation of 0.5



(c) Physical/virtual world correlation of 1

Figure 7.3 Comparison in interactive delays between distributed proxies and distributed locale servers when varying the number of server.

used, the distributed proxy architecture has slightly lower delay, especially when the correlation is above 0.6.

We repeat a similar experiment as the previous one except potential servers in this case are selected from ISP POPs. This is to address the fact that proxies are often deployed at network edges. We expect this scenario is more favorable for the distributed proxy architecture. In the distributed proxy architecture, we use both simple heuristics and greedy heuristics for proxy assignment. The results are shown in Fig. 7.2. In the distributed proxy architecture, using greedy heuristics for proxy assignment results in smaller delay than the simple heuristics in all cases. The distributed proxy architecture using greedy heuristics also had equal or smaller delay than the distributed locale server architecture in all cases. It is observed that both of the distributed proxy architectures provide lower delay than the distributed locale server architecture when there are a large number of servers. However, when the number of servers is small (one twentieth the number of POPs), the distributed locale server architecture has better delay performance than distributed proxy using simple heuristics in this case. This is due to the fact that the gap between the simple heuristics and the greedy heuristics is large when only a small number of servers are used. The distributed proxy and distributed locale server architectures have relatively similar delay performance when medium number of servers (one fifth the number of POPs) are used. While distributed proxies perform better than distributed locale servers when there are a large number of servers (half the number of POPs).

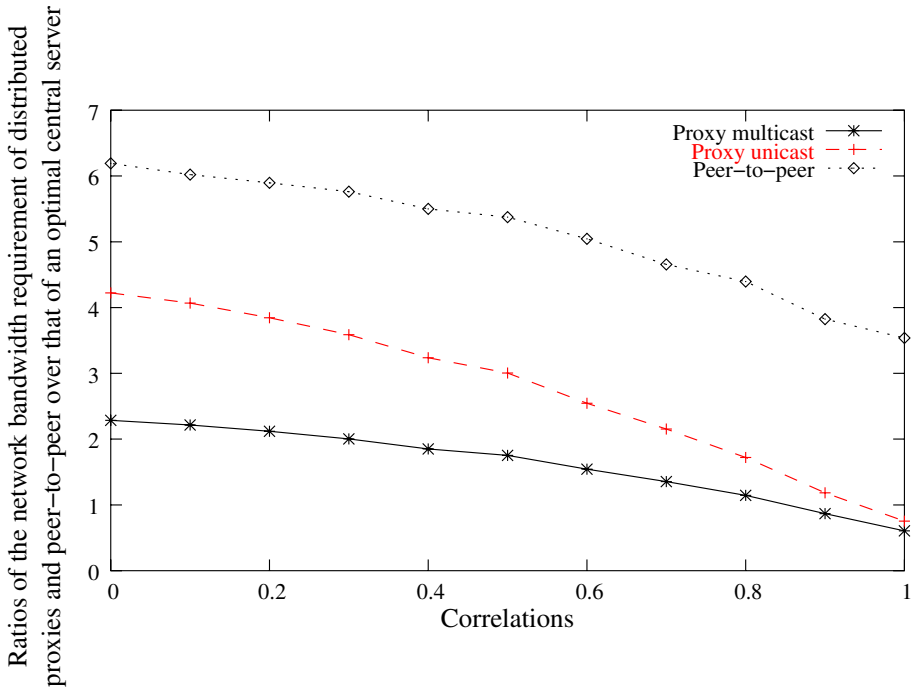
In the third experiment, we compare the delay performance of distributed proxies and distributed locale servers when varying the number of servers. Fig. 7.3 shows the interactive delay of these architectures when varying the number of servers at physical/virtual world correlation parameters of 0, 0.5 and 1. As indicated in the figure, at no correlation, distributed locale servers have similar interactive delay to the central server and increasing the number of servers does not improve the delay performance. On the other hand, at no correlation, when only five servers are used, the interactive delay of distributed proxy architecture is about 20% higher than the central server's interactive delay. This delay is 20% lower than the central server's delay when fifty servers are deployed. At correlation of 0.5, the delay of distributed

locale server is reduced by about 15% when the number of servers increases from 5 to 50. The delay of the distributed proxy architecture reduces from 10% above the central server's interactive delay to about 30% below that delay. At correlation of 1, the interactive delays of both architectures are about 50% lower than the central server's interactive delay when the number of server exceeds 20. In addition, the distributed proxy architecture has smaller delay than distributed locale server in this case.

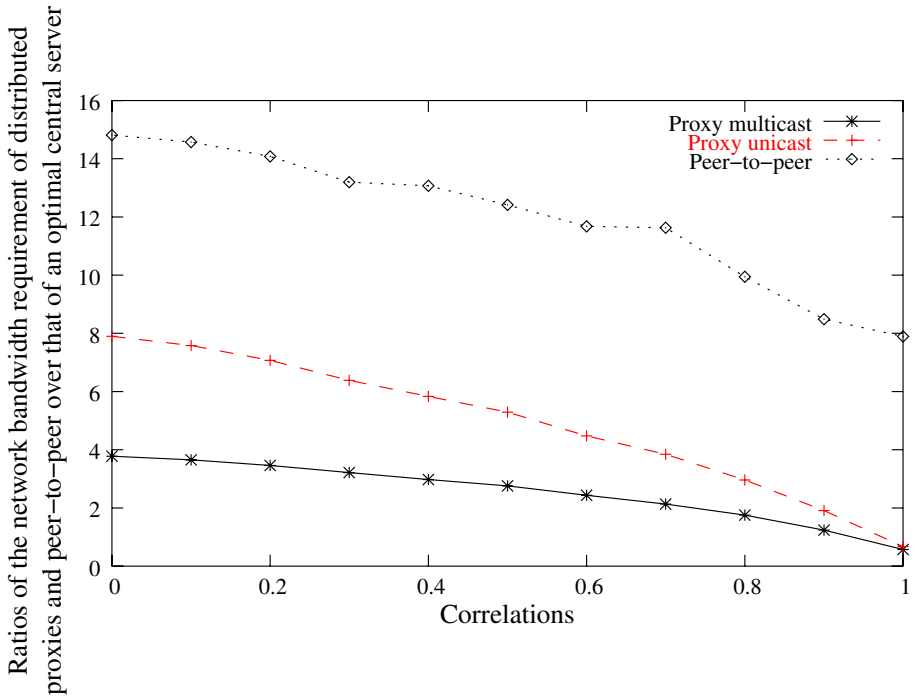
It is shown that increasing the number of servers reduces delays in both architectures. However, this effect on each architecture is different. Due to our optimization algorithm, the distributed locale server architecture always achieves equal or smaller delay than the central server, regardless of the number of servers. In distributed locale servers, when the number of servers goes above certain value, approximately one fifth of the number of POPs, further improvements in the interactive delays are very small. The delay of distributed proxy architecture depends considerably on the number of servers. When the number of servers is small (under one tenth the number of POPs), distributed proxy could have higher delay than the central server due to large distances between some proxies and POPs.

7.3.3 Network Bandwidth Usage

In the following simulations, we aim to evaluate the network bandwidth usage of different architectures. While the interactive delay of each delivery architecture does not depend on game player grouping behaviors, the network bandwidth usages depend considerably on these behaviors. Fig. 7.4 shows the network bandwidth usages of proxy multicast, proxy unicast and peer-to-peer in clan/crowd based games. In all these architectures, the network bandwidth usage decreases as the correlation is increased. This is expected since the higher the correlation is, the fewer audio flows are needed across the network, and less resources are used. The results indicate that distributed proxies reduce the core network bandwidth usage of the peer-to-peer architecture, especially at high correlation. However, at low correlation, the proxy unicast still uses up to eight times the network bandwidth of the central server in a crowd based games. Network bandwidth usages are smaller in clan based games due



(a) Clans



(b) Crowds

Figure 7.4 Network bandwidth usages of distributed proxies in clan/crowd based games when varying the physical/virtual world correlation.

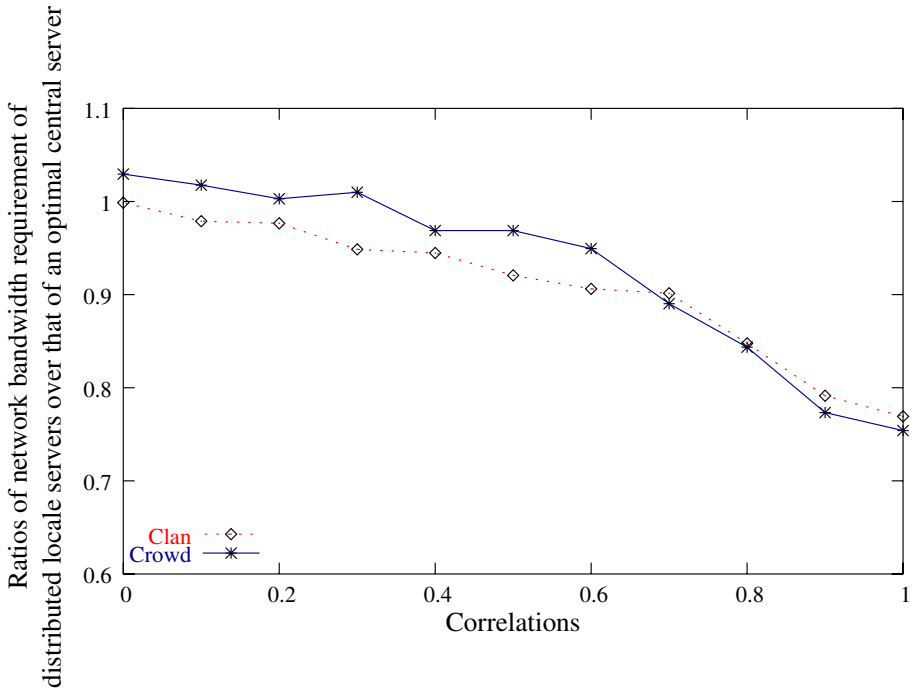


Figure 7.5 Network bandwidth usages of distributed locale servers in clan/crowd based games when varying the physical/virtual world correlation.

to smaller numbers of avatars in hearing range. Multicast only reduces the network bandwidth usages of proxy unicast by up to 50%. With the implementation of network multicast, distributed proxy architecture still uses four times the bandwidth of the central server.

On the other hand, as indicated in Fig. 7.5, the distributed locale server architecture uses relatively equal or less network bandwidth resources than the central server architecture. At low correlation, the distributed locale server architecture has relatively similar network bandwidth usage compared to the central server. When the correlation is increased to 1, the bandwidth usage of the distributed locale server architecture is reduced by more than 20%.

Furthermore, the network bandwidth usage of the distributed locale server architecture does not increase as avatar density increases. This is the distinct advantage of

the distributed locale server architecture over the distributed proxy architecture. As indicated in Fig. 7.6a, in a loner based game, while the network bandwidth usages of proxy architectures increase almost linear with avatar density, the network bandwidth usage of distributed locale server remains at relatively the same level with the central server. The network bandwidth usage of the distributed proxy architecture and peer-to-peer are low in this case due to small numbers of avatars in hearing range. As indicated in Fig. 7.6b, in a clan/crowd based game, the network resource requirements of the distributed proxy and peer-to-peer architectures increase significantly as avatar density increases. However, the network bandwidth usage of the distributed locale server architecture is still relatively similar to that of the central server.

7.4 Summary of Results and Recommendations

In this chapter, we have compared characteristics of different delivery architectures. The chapter also provides simulation experiments to investigate the performance of two 'dual' distributed server architectures in a range of avatar grouping behaviours and server distribution scenarios. This Section presents an overall performance summary of delivery architectures. This summary also includes some results from simulation experiments in previous chapters (from Chapter 4 to Chapter 6).

7.4.1 Impact of Avatar Aggregations on Delivery Architectures

Avatar aggregation behaviours have insignificant impact on network bandwidth usage of the central server architecture and distributed locale server architecture. The network bandwidth usage of central server is low and does not increase with avatar density. Since a central server sends a maximum of fixed number of audio streams (e.g, 2 streams in our simulations) to each participant, when the number of avatars in hearing range exceeds this number, the number of voice streams are unchanged. The network bandwidth usages of distributed locale servers are equal or less than the central server, depending on the correlation.

On the other hand, the network bandwidth usage of peer-to-peer and distributed proxies increases linearly with avatar density. These usages are low and not much differ-

ent from the central server in loner based virtual world but increase tremendously in clan/crowd based virtual worlds. In particular, the network bandwidth usage of peer-to-peer and distributed proxies can increase up to 20 and 15 times of the bandwidth usage of central server, respectively.

7.4.2 Impact of Number of Servers

The performance of distributed locale servers and distributed proxies are dependent on the number of available servers. In distributed locale server, the interactive delay goes down as the number of servers increases. This delay is similar to the central server's delay when a small number of servers is used but is reduced by up to 60% when a large number of servers is used. However, the incremental delay improvement is insignificant when the number of servers exceeds certain values, approximately one fifth the number of POPs.

On the other hand, the number of proxies has a significant impact on the latency. When only a small number of proxies available (e.g., less than one tenth of the number of POPs) distributed proxies can even have higher delay than the central server. This delay is reduced as the number of proxies increases and drops by nearly 50% when proxies are available at all POPs.

7.4.3 Impact of Correlation

Virtual/physical world correlation does not have any impact on the central server but does have significant effect on the performance of all other architectures. When the correlation increases, communication between players in the virtual world are among physically closer participants. Hence, both the network bandwidth usage and the interactive delay of these architecture are reduced.

In particular, the interactive delay of each of distributed server architectures is reduced by up 60% when correlation is increased from 0 to 1. In this situation, the network bandwidth usage of distributed locale servers is reduced by up to 40%. Correlation has more impact on the network bandwidth usage of peer-to-peer and distributed proxies. Specifically, when the correlation increases from 0 to 1, the band-

width usage of peer-to-peer is reduced by 70% and 80% in crowd and clan based virtual world, respectively.

7.4.4 Efficiency of Multicast

There are two attributes that affect the efficiency of multicast: avatar density and the spread of game participants. Multicast is effective only when the average number of avatars in hearing ranges is large (i.e crowds). In these scenarios, when game players are connected to a small number of ISP POPs, the bandwidth cost saving of multicast is significant. However, when game players are widely spread in the Internet, the efficiency of multicast is reduced.

The proxy unicast architecture also has lower bandwidth usage than the peer-to-peer architecture, especially when game players are connected to a small number of POPs. Although this bandwidth usage reduction is not as significant as multicast, the proxy unicast is worth considering since network multicast is not available Internet wide.

7.4.5 Discussions on Choice of Delivery Architectures

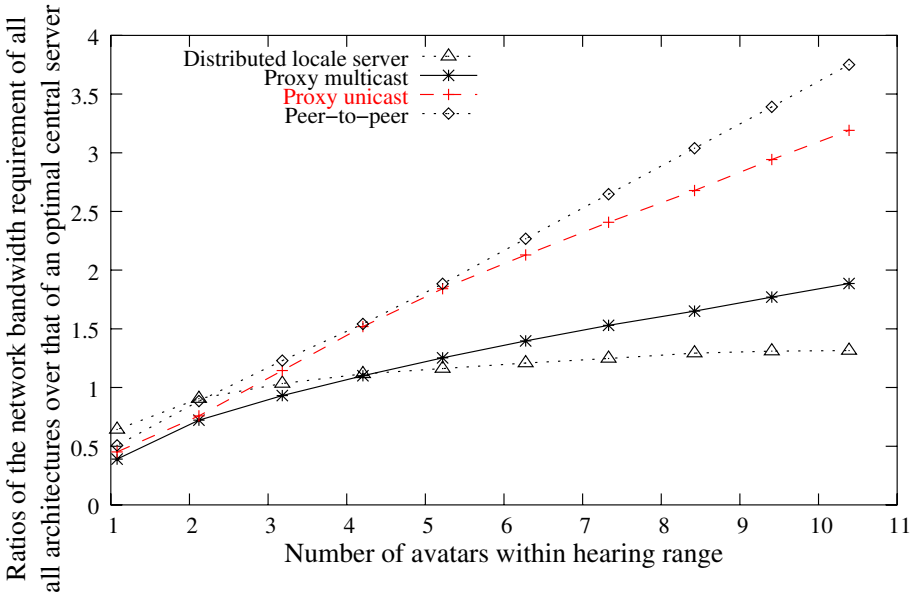
When all architectures are taken into consideration, the following suggestions are presented:

- For loner based games, the peer-to-peer architecture would be favored due to low delays in direct paths between avatars, however, security/anonymity issues must be addressed.
- For games consisting of crowds and clans, both distributed server architectures are suitable, especially, when there is a correlation between the virtual world and the physical world. In this case, if delay is the key constraint, the distributed proxy architecture is favored. If bandwidth is the key constraint, the distributed locale architecture is more suitable.

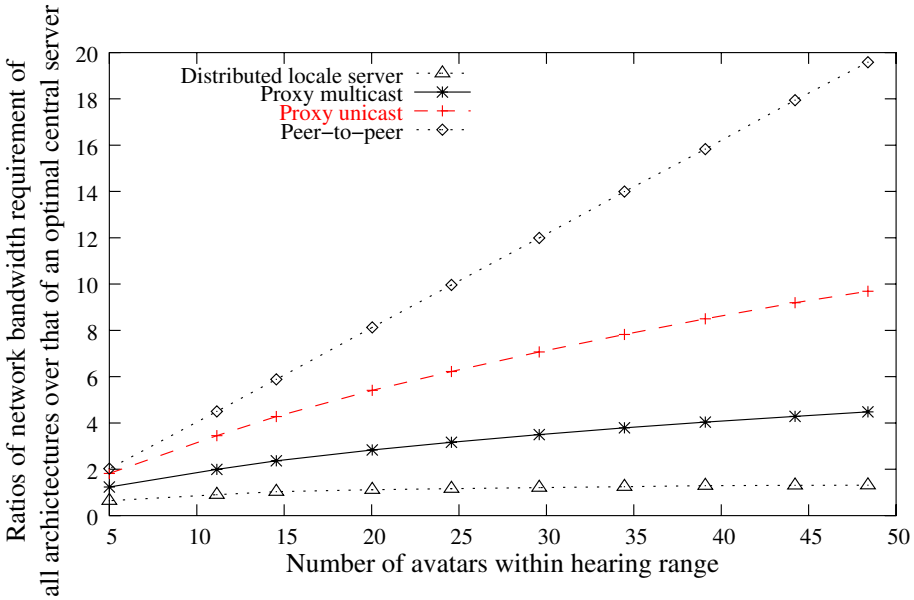
With regard to the two ‘dual’ distributed server architectures, our results indicate that if a large number of servers are available over the Internet, both of these architectures are cost-effective since they reduce the delay of the central server architecture. This

scenario is possible since ISPs are generally willing to put servers in their networks in order to attract more game players (or Internet service subscribers) and increase their revenues. Furthermore, if a small number of servers are available, distributed locale server would be the choice while distributed proxy architecture is more suitable when a large number of servers are available. In addition, the distributed locale server architecture has better bandwidth efficiency than the distributed proxy architecture, especially in games that have high avatar density.

In many games, where all different player characteristics may be apparent in different parts of a virtual world, a hybrid architecture should be considered, in which, parts of the virtual world that mainly consist of loners would use the peer-to-peer architecture, while crowds or clans would either use one of these distributed server architectures. Also, this architecture requires a dynamic signalling of a hybrid system, and the evaluation of this architecture is future work.



(a) Loners



(b) Clans and Crowds

Figure 7.6 Effect of avatar density on network bandwidth usages of different architectures

Chapter 8

Server Processing Resource Management

8.1 Introduction

In previous chapters, we have evaluated different architectures to provide the immersive voice communication service to MMOG using two basic performance metrics: network bandwidth costs and delays. However, the delay metric does not take into account processing delays at audio mixing servers. While efficient server assignment and adequate bandwidth provisioning can minimize propagation delay and network transmission delay, the audio mixing delay is strictly correlated to the QoS management at each server. Minimizing server processing delay is essential for the immersive audio communication service.

In Chapter 2, we have reviewed various architectures that can enable real-time processing of packets in the network. This type of processing enables contents to be created or manipulated in real-time for end-user requirements. In these architectures, network processors or server farms are attached to network nodes in order to provide computation for a number of applications. These applications are referred to as *real-time content creation applications*. These applications include network-based processing applications such as transcoding, encryption as well as the proposed immersive audio mixing application.

These applications need to share the server processing capacity at network nodes since it would be expensive to deploy a single server for each application in a large number of network locations. SWON (Boustead et al., 2004) architecture is particularly suitable for this purpose. By using SWON, a service provider only needs to reserve certain capacity at the server farm in the form of *virtual servers* and install required software and configure that virtual server to support its application. We envisage that the deployment of the immersive audio services for a number of network games will significantly benefit from using SWON.

In this chapter, we introduce a processing resource scheduling algorithm called Start-time Weighted Fair Queueing (SWFQ). As discussed in previous chapters, the delay performance of distributed locale server and distributed proxy architecture will improve when there are a large number of servers. Deploying a large number of servers at various location in the Internet for a single game would be very expensive. Using SWFQ, each server can be effectively shared between different real-time content creation applications. As shown in Fig. 8.1, a service provider can deploy a server infrastructure using SWON and use SWFQ to provide processing resource sharing for immersive audio scene creation among several different games. Once a server assignment algorithm chooses a suitable set of server sites, processing resource reservation is made at each server site based on the number of players assigned to that server. In addition, due to movement of avatars and player joining or leaving the game, the processing resource requirement of an audio scene creation at a server may change. This change can be addressed by renegotiating resource reservation and assigning a new scheduling weight for the application.

This chapter is concerned with designing an efficient processing resource scheduler at a server farm to minimize delays and ensure fair processing resource allocation between applications that require real-time creation of content. In Section 8.2.1, we first briefly describe requirements of the immersive audio mixing operations at a server and discuss several approaches for processing resource management for applications that require real-time content creation. In Section 8.3, we discuss the design of SWFQ. Section 8.4 presents an analysis of this algorithm. Finally, in Section 8.5, simulation experiments are carried out to evaluate the performance of this algorithm

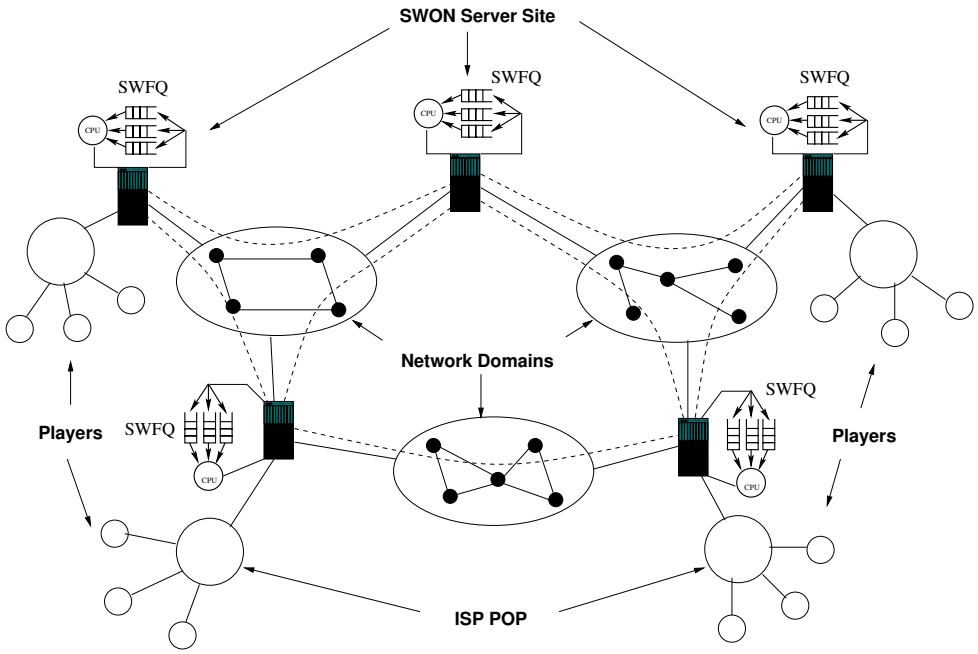


Figure 8.1 An example of implementation of SWFQ in SWON.

in a wide range of applications. Section 8.1 gives an implementation scenario of SWFQ and Section 8.6 concludes.

8.2 Server Processing Resource Management

8.2.1 Processing Resource Requirements of Immersive Audio Mixing Operation

Fig. 8.2 shows an outline of an immersive audio mixing processor. Packets arriving from different audio streams are buffered. For each computation time interval, a packet in each stream is selected for mixing to produce the audio scene. The computing requirements will depend on the number of players in the game which is strictly correlated to the number of audio streams sent to the processor. Even when the number of players is unchanged, the computation requirements for audio scenes are dependent on several factors: 1) Number of audio packets successfully reaching the server; 2) Positions of avatars which determine what mixing operations (e.g.,

additions and multiplications) are performed on these packets. 3) Contents of these packets depending on state of conversation (e.g., packet representing silence state or talk spurts).

Researchers in our research group have implemented an immersive audio mixing server for several games including Quake 3 and Wolfenstein Enemy Territory. We carried out an experiment on a Quake 3 game of five participants and used a Celeron 1.2 GHz server for computing immersive audio scenes for these players. In this experiment, an ADPCM codec with 32kbps data rate is used. The server buffers audio packets from players and computes an audio scene at every 32 ms interval and sends the results back to each player (Boustead et al., 2005). At each interval, the relative position of each avatar used for computing audio scenes and a server processing time is recorded. Fig. 8.3 shows the server processing times of computing audio scenes for the game during 5000 intervals or 160 seconds of the game. The computation times vary significantly from 1.7 ms to 5.5 ms.

These variances are due to movements of avatars and conversation patterns. For example, when avatars are close to each other, the numbers of avatars in hearing ranges increase, and the computation requirement for that audio scene also increases. Even when there is no movement of avatars, the computation requirement also depends on the conversation patterns such as talk spurt or continuous chatter which determines the number of audio packets presenting actively speaking state delivered to the server at each interval. In massive multiplayer online games, the processing requirement of audio mixing for the game may have similar behavior.

8.2.2 Server Processing Resource Management Model

8.2.2.1 Classification of Applications

The basic model of a processing resource scheduler at a server site (e.g, a SWON server site) is shown in Figure 8.4. In this model, a processing server is connected to a router or switch. Packets belonging to different real-time content creation applications are delivered to the server. These packets are buffered and placed in a queue for processing at the processor scheduler. After being processed, they come back to

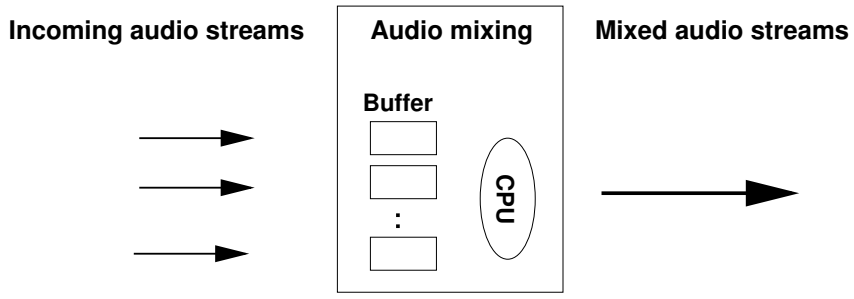


Figure 8.2 Immersive Audio Mixing.

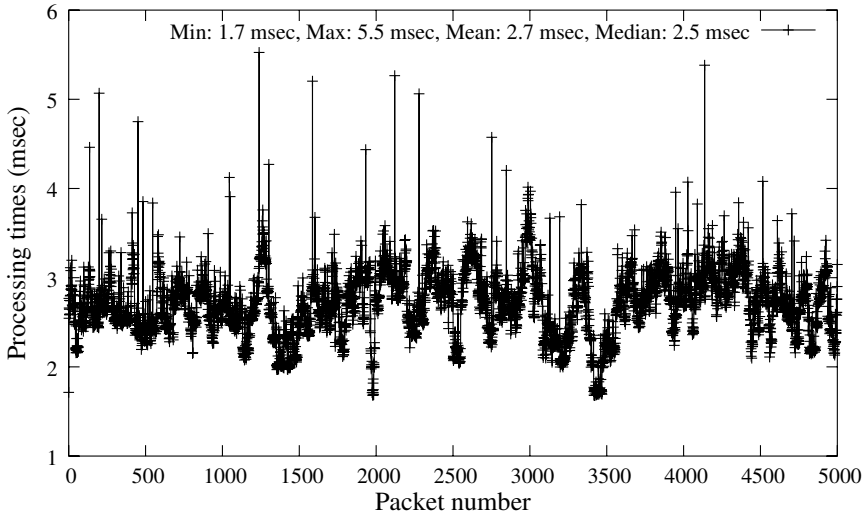


Figure 8.3 Variation in actual processing time requirements of immersive audio scene of a Quake 3 game with five players.

the router/switch for transmission.

Applications that require real-time processing of content can be classified into two classes: header processing applications and payload processing applications. In header processing applications, processing only involves read and write operations of packet headers. Examples of header processing applications are IP forwarding, transport layer classification and QoS routing. On the other hand, in payload processing applications, read and write operations can be performed on packet payload. Exam-

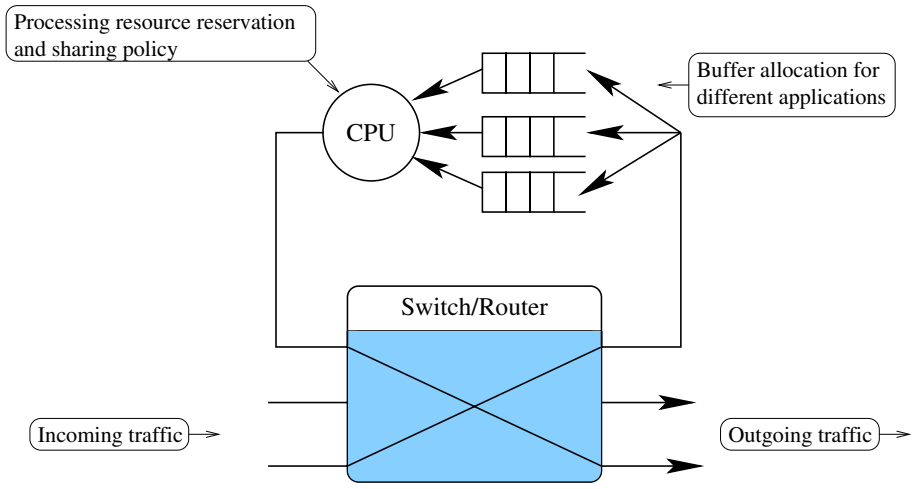


Figure 8.4 Model of a node processing resource management.

ples of payload processing applications are packet compression, packet transcoding, encryption and decryption. Immersive audio mixing is also a payload processing application. For this application, the server processes the contents of all audio packets arrived from players and computes individual audio scene packets for each player.

8.2.3 Processor Resource Scheduling

In Section 2.4.1.3, we reviewed several techniques and approaches for server resource management. Depending on the type of applications running on a server, each approach has a different way of managing processing resource. We also discussed the problem of sharing processing resource among different content creation applications. Current research efforts aim to design suitable packet-based processing resource schedulers for these application instead of time sharing as in traditional operating systems.

The server node architectures in (Campbell et al., 1999) (Peterson, 2001) enforce isolation of packet processing between flows or threads for security and accounting purposes. However, QoS scheduling and admission control are not specified. The work in (Qie et al., 2001) discusses the problem of scheduling computation resources in a software-based router but relies on the assumption that processing times

of packets are pre-determined. Galtier *et al.* (Galtier et al., 2001) shows that the prediction of packet execution times in an experimental test bed can have large mean errors.

The research in (Pappu and Wolf, 2002) shows that the processing load of some networking applications are predictable and correlated to the packet sizes. This observed correlation is then used for specifying processing resource reservations, and scheduling based on estimation of each packet processing times. However, in general purpose processing, it is hard to determine the processing time of an arbitrary packet.

In addition, processing load and operating system scheduling of a server can have some impact on packet processing time. Experiments in (Sabrina and Jha, 2003) on three payload processing applications, MPEG2 Encoder, RC2 Encryption and RC2 Decryption running on a Linux server shows that processing times can vary although the data sizes are the same for all executions. As an example, Fig. 8.5 shows the variation in actual processing times of a MPEG2 data block of 24576 bytes in 1000 repetitive executions on a Pentium 4, 1.8 GHz processor. In this experiment, there is no other process running except those belonging to operating system processes. It is demonstrated that the processing times of identical frames can be different due to operating system process scheduling. In addition, as indicated in Section 8.2.1, processing requirements of computing audio scenes can vary significantly even when the number of players is unchanged.

In short, in real-time content creation applications, it is generally hard to determine execution times of packets for scheduling. In the next section, a processing resource scheduling algorithm is introduced to address this problem.

8.3 Start-time Weighted Fair Queueing

One of the key problems for scheduling processing resource is the inability to determine execution times of packets from information in headers for scheduling. Therefore, it is difficult to maintain the fair share of processing resources among applications and provide bounds on the processing latency. In bandwidth resource schedul-

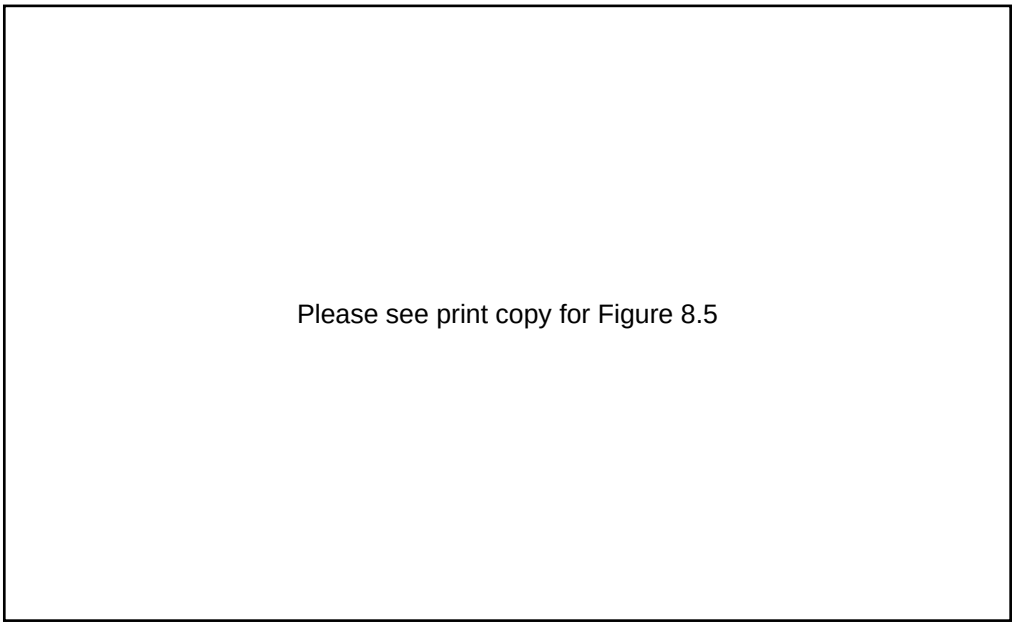


Figure 8.5 Variation in actual processing time of MPEG2 data block of fixed length for different executions (Sabrina and Jha, 2003).

ing, most fair scheduling schemes rely on obtaining the packet length in the packet header to determine the scheduling decision among different sessions. In processing resource scheduling, each packet carries codes, which can take different times to process at servers.

Traditional operating systems use time sharing and context switching for processing resource sharing. However, in network processing, the smallest unit of processing is a complete packet. In many cases, context switching cannot be considered because saving and recovering processing state will cause large processing overhead. In addition, erroneous or malicious packets can overuse CPU resources, therefore, penalize other applications and reduce CPU performance considerably.

In this section, we first review basic packet scheduling disciplines that are used in traditional network for bandwidth scheduling. Then, we introduce SWFQ and show that it offers good fairness and delay properties without the knowledge of each packet processing time in advance for scheduling. We propose to use this scheduling algorithm to concurrently support a wide range of applications including:

- An application that require QoS, such as a bound on latency and an average

processing rate, can do *processing resource reservation* based on an estimated average processing requirement of that application.

- Other applications that do not require QoS guarantees, can receive fair processor sharing on a *best-effort* basis.

8.3.1 Packet Scheduling Disciplines in Traditional Networks

This section briefly describes the fundamental design of packet scheduling disciplines in traditional networks. Generalized Processor Sharing (GPS) (Parekh and Gallager, 1993) is an ideal scheduling discipline based on a fluid flow model in which the traffic is infinitely divisible and each session is serviced simultaneously according to its weight. In packet-switched networks, however, the minimum service unit is a packet and only one traffic stream can receive service at a time. Therefore, most packet fair queuing (PFQ) algorithms are based on approximating GPS.

PFQ algorithms often simulate GPS by assigning virtual finishing times for packets as if GPS was running. Packets from different sessions are placed in different logical queues and stamped with virtual finishing times. Packets from the head of session queues will be selected to be scheduled based on the increasing order of the virtual finishing times. Computing virtual finishing times requires the size of packets from packet headers. Most PFQ algorithms have the same way of updating virtual finishing times of packets (p_i^k) based on the system virtual time, denoted as $V(t)$, as follows.

$$F_i^k = max\{F_i^{k-1}, V(a_i^k)\} + \frac{L_i^k}{\phi_i} \tag{8.1}$$

where:
p_i^k: Packet number k in session i.
ϕ_i: Weight of session i.
F_i^k: Virtual finishing time of packet p_i^k.
a_i^k: Arrival time of packet of packet p_i^k.
L_i^k: Length of packet p_i^k.

The virtual finishing time F_i^k represents a normalized amount of service, with respect to its share, that session i has received right after packet k is served. When a PFQ

algorithm schedules packets in increasing order of virtual finishing times, it aims to equalize the normalized amount of services of all backlogged sessions. The role of the system virtual time is to compute virtual finishing times of packets arriving at unbacklogged sessions in order to equalize the normalized services of these sessions with current backlogged sessions. The differences in defining system virtual time functions in packet fair queueing algorithms result in different implementation complexities, fairness measurements, and delay behavior.

Basically, there are two major approaches for updating the system virtual time.

- *Simulating Generalized Processor Sharing (GPS):* This approach is used in Weighted Fair Queuing (WFQ) (Demers et al., 1990), Packet Generalized Processor Sharing (PGPS) (Parekh and Gallager, 1993), and Worst-case Fair Weighted Fair Queuing + (WF^{2}Q+) (Bennett and Zhang, 1997). These schemes use or estimate the system virtual function $V_{GPS}(t)$ that reflects the normalized service each backlogged session should receive under GPS by time *t*. For example, in WFQ and PGPS, the system virtual time is defined as

$$V(t+\tau) = V(t) + \frac{\tau}{\sum_{i \in B(t)} \phi_i} \tag{8.2}$$

  where $B(t)$ is the set of current backlogged sessions at time t. In this equation, the slope of the system virtual time is inverse proportional to the number of backlogged sessions, therefore, it allows sessions to receive more service when the number of backlogged sessions decreases, and vice versa.

- *Self-clocking:* This approach is used in Self-Clock Fair Queueing (SCFQ) (Golestani, 1994) and Start-time Fair Queueing (SFQ) (Goyal et al., 1997). This approach has low complexity of maintaining the system virtual time but experiences higher worst case delay.

8.3.2 Server Processing Resource Scheduling Model

From equation 8.1, if we define start time $S_i^k = max\{F_i^{k-1}, V(a_i^k)\}$, it can represent the amount of service, normalized to its service share, session i has received before

packet k is served. If a server chooses to schedule packets in increasing order of start times, it should be able to provide equivalent performance to scheduling using finishing times. Therefore, we design a PFQ algorithm based on start times since it would not require packet lengths in advance. For scheduling based on finishing times or start times, the system virtual time is important to compute finishing or start times of packets arriving at unbacklogged sessions. We present a general methodology to design PFQ algorithms based on start times as follows.

Consider a model of a processor scheduler serving N sessions which have packets with different execution time requirements. Due to unknown processing requirements of packets, only one pair of S_i and F_i is maintained for each session queue i. When packet p_i^k arrives at the head of session i queue, the start time is computed as follows.

$$\begin{cases} S_i = max\{F_i, V(a_i^k)\} & \text{if session } i \text{ queue is empty} \\ S_i = F_i & \text{otherwise} \end{cases} \tag{8.3}$$

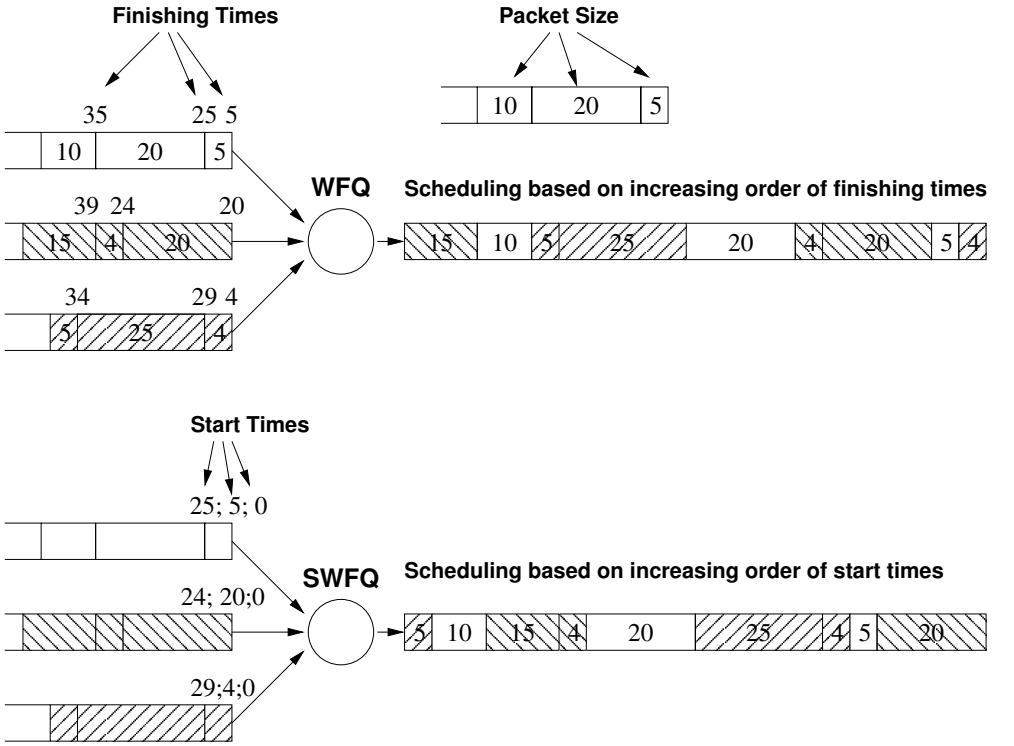
The finishing time F_i is updated only when this packet, which has processing cost P_i^k, has been processed.

$$F_i \;=\; S_i + \frac{P_i^k}{\phi_i} \tag{8.4}$$

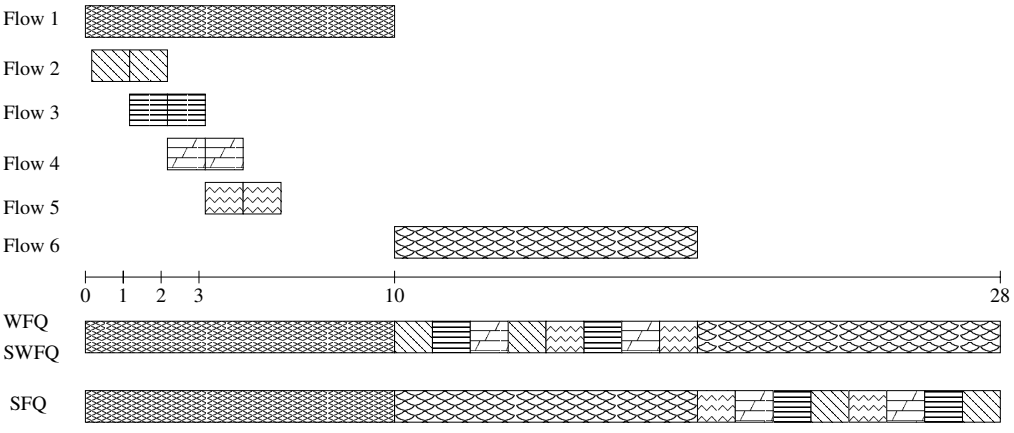
This finishing time is then used to compute the start time of the next packet in session i as shown in equation 2. In order to avoid malicious packets that can overuse CPU resources, we define P_i^{max} as the maximum allowed processing size of packets in session i.

Using this model, we propose a PFQ algorithm called Start-time Weighted Fair Queueing (SWFQ). SWFQ uses the same system virtual time used in WFQ, but schedules packets in increasing order of start times. SWFQ is a generalization of a scheme called Fair Queueing based on starting time (FQS) which is introduced in (Greenberg and Madras, 1992) and is based on uniform processor sharing.

One of the existing PFQ algorithms in the literature that can also be used for processing resource scheduling in this model is Start-time Fair Queueing (SFQ) (Goyal et al., 1997). In SFQ, the system virtual time is defined as the start time of the current packet in service and is updated each time a packet begins service. The main attrac-



(a)



(b)

Figure 8.6 Example of WFQ, SWFQ and SFQ.

tive feature of SFQ is the simplicity of maintaining the system virtual time. However, one problem with SFQ is that the system virtual time is constant for some period of time if packets from backlogged connections have the same start times. During this time, if some new connections become backlogged, the start times of packets belonging to these connections will be set to equal to start times of packets that already backlogged in this period. This may result in larger delay in processing packets from already backlogged sessions. This behavior can be seen in the analysis carried out by Bennett and Hui Zhang in (Bennett and Zhang, 1997) and the example shown in the next section.

8.3.3 Example of SWFQ

In this section, we provide examples to compare basic operations of SWFQ with WFQ and SFQ. Fig. 8.6a shows how WFQ and SWFQ compute finishing times and start times to enable fair service ordering. In WFQ, the packet size is known, hence the first packets of these three flows are assigned with appropriate finishing times based on packet sizes. These finishing times are 5, 20, and 4, respectively. Hence, these packets are scheduled in an increasing order of finishing time which is the same to the service finishing order in GPS. On the other hand, in SWFQ, due to unknown processing time, the start times of the first packet of these three flows are set to 0. In the worst case, SWFQ may misorder the first packet of a busy session compared to the service finishing order in GPS (the service processing order of the first three packet should be 4-5-20 instead of 20-5-4). Therefore, SWFQ has different properties compared to WFQ that we will analyze in the next session.

In SFQ, the system virtual time is defined as the start time of the current packet in service and is updated each time a packet begins service. Therefore, the system virtual time will stop advancing for a period of time if a burst of packets arrives at empty queues in a short interval, resulting in larger delays to packets from already backlogged sessions. This behavior is demonstrated in Fig. 8.6b, which shows the arrival sequences and service orders of packets in six flows, each reserving the same rates ($\phi_i = 1/6$). The server capacity is 1 unit/s; the costs of packets p_1^1 and p_6^1 are 10 units; the costs of other packets are 1 unit. At time $t = 0$, p_1^1 arrives, followed by

p_2^1 just after $t = 0$. Packets p_3^1, p_4^1, p_5^1, and p_6^1 arrive at time $t = 1$, $t = 2$, $t = 3$, and just before $t = 10$, respectively.

In SWFQ, during the processing time interval of p_1^1, the system virtual times increases from $V(0) = 0$ to $V(10) = 14.9$, and the start times of p_2^1, p_3^1, p_4^1, p_2^2, p_5^1, p_3^2, p_4^2, p_5^2, and p_6^1 are 0, 3, 5, 6, 6.5, 9, 11, 12.5, and 14.9, respectively. Consequently, the service order of SWFQ is similar to WFQ, in which the service of each session finishes in the same order as if GPS was running. In SFQ, the system virtual time remains unchanged when p_1^1 is in service. Therefore, p_2^1, p_3^1, p_4^1, p_5^1, and p_6^1 have the same start times equal to 0 and are selected arbitrarily. As shown in Figure 2b, in the worst case, SFQ can misoder packet considerably, resulting in large delays to early arrived packets such as p_2^1. We will show later in simulations that SWFQ can improve the delay behavior of SFQ due to its more accurate system virtual time.

8.4 Analysis of SWFQ

8.4.1 Fairness Analysis

In this section, we analyze the throughput of SWFQ compared with WFQ. We will show that SWFQ has a fairness equally comparable to WFQ. Since WFQ is widely known as a fair packet scheduling algorithm in the literature, it is proven that SWFQ is also a fair scheduling algorithm.

Theorem 1 *For all time t and sessions i,*

$$\hat{W}_i(0,t) - W_i(0,t) \leq P^{max} \tag{8.5}$$

where:
$\hat{W}_i(0,t)$: Work done in the SWFQ server in the time interval $(0,t)$.
$W_i(0,t)$: Work done in the GPS server in the time interval $(0,t)$.
P^{max}: Maximum processing size of a packet in the system.

This theorem shows the maximum amount of service by which a session under SWFQ can exceed GPS. Because GPS and SWFQ are both work-conserving, they

have the same busy period. Therefore, it is sufficient to prove the result for each busy period. Without loss of generality, we prove the theorem in the Appendix A.1 with the assumption that session i is constantly backlogged in interval $(0, t)$.

Theorem 2 *For all time t and sessions i,*

$$W_i(0,t) - \hat{W}_i(0,t) \leq min((N-1)P^{max}, r_i \max_{1 \leq n \leq N}(\frac{P_n}{r_n})) \tag{8.6}$$

where:

r: Server rate.

r_i: Reserved rate of session i.

P_n: Processing size of a packet in session n.

N: Number of sessions sharing the server.

The theorem states the maximum amount of service that a session under SWFQ can lag GPS. The proof is given in the Appendix A.2.

It is interesting to see that the bound on the throughput discrepancy of SWFQ and WFQ compared to GPS are the same but in opposite directions. From the result of the analysis in (Rexford et al., 1995), WFQ can lead GPS by the same amount as SWFQ can lag GPS in Theorem 2. In addition, from the result of Theorem 1, SWFQ can lead GPS by the same amount that WFQ can lag GPS in (Parekh and Gallager, 1993). Therefore, SWFQ has *comparable fairness* to WFQ.

8.4.2 Delay Analysis

8.4.2.1 Delay Guarantee

In this section, we analyze the delay bounds on packets in different application flows in SWFQ.

Theorem 3 *For all sessions i,*

$$L_{SWFQ}(p_i^k) \leq EAT(p_i^k) + \sum_{n \in B(t), n \neq i} \frac{P_n^{max}}{r} + \frac{P_i^{max}}{r} \tag{8.7}$$

where:

$L_{SWFQ}(p_i^k)$: Delay guarantee for packet p_i^k.

$EAT(p_i^k)$: Expected arrival time of packet p_i^k.

$EAT(p_i^k)$ is defined as

$$EAT(p_i^k) = max\{a(p_i^k), EAT(p_i^{k-1}) + \frac{P_i^{k-1}}{r_i}\} \tag{8.8}$$

where $EAT(p_i^0) = -\infty$.

The delay guarantee for packet p_i^k is the bound on the departure time of that packet based on its *expected arrival time* (EAT) (Goyal and Vin, 1997) (Xie and Lam, 1995). The proof of this Theorem is given in the Appendix A.3.

For WFQ, this delay guarantee, given in (Goyal et al., 1997), is,

$$EAT(p_i^k) + \frac{P_i^{max}}{r_i} + \frac{P^{max}}{r}$$

Note that, since the delay guarantee here is independent of the traffic source specification, it is not the delay bound. If session rates are controlled, e.g. by a leaky bucket, the delay bound can be derived from this delay guarantee (Xie and Lam, 1995) (Goyal et al., 1995).

The attractive feature of PFQ using finishing times like WFQ is the ability to guarantee a delay to a session flow based on the flow's properties such as the reserved rate r_i and the maximum packet processing size P_i^{max}. On the other hand, the delay guarantees of SWFQ depend on the maximum processing size of packets in all backlogged sessions at the server. In some situations, SWFQ can provide smaller delay guarantees than WFQ. For example, consider session i that has larger P_i^{max} compared with other sessions and all sessions reserve the same rate. It follows that,

$$\sum_{n \in B(t), n \neq i} (\frac{P_n^{max}}{r}) < \frac{P_i^{max}}{r_i} \tag{8.9}$$

As a result,

$$L_{SWFQ}(p_i^k) < L_{WFQ}(p_i^k) \tag{8.10}$$

In short, although SWFQ does not have the attractive delay guarantee feature of WFQ, it can provide predictable delay guarantees which can be acceptable in the context of processing resources scheduling. For real-time content creation, this delay bound is a key for minimizing processing delay as well as delay jitter at a server.

8.4.2.2 Delay Bound

The delay bound can be derived from the delay guarantee under some assumption of packet processing requirements of a flow and packet arrival process. Let $A(p_i^k)$ be the actual arrival time of packet p_i^k. We assume that actual processing requirements of packets in flow i conforms to the following arrival function $AP(t_i, t_2)$:

$$AP(t_1, t_2) \leq \delta_i + r_i(t_2 - t_1) \tag{8.11}$$

where:
δ_i: Maximum burst (in term of processing requirement).
The delay bound is determined as follows.

Theorem 4 *For all sessions i, the delay bound is*

$$L_{SWFQ}(p_i^k) - A(p_i^k) \leq \frac{\delta_i}{r_i} - \frac{p_i^k}{r_i} + \sum_{n \in B(t), n \neq i} \frac{P_n^{max}}{r} + \frac{P_i^{max}}{r} \tag{8.12}$$

The proof of this Theorem can be found in Appendix A.4

8.5 Simulation Experiments

8.5.1 Simulation Setup

In order to demonstrate the properties of SWFQ, we have modified the Network Simulator version 2 (ns-2) (Fall and Varadhan, 2002) to implement SWFQ and SFQ, and to support simulations of several real-time content creation applications.

In the first and second experiment, we simulate three network processing applications that are defined in (Pappu and Wolf, 2002): IP Forwarding (header processing) with very low processing cost per packet; Cast Encryption with medium processing

cost per packet; and Forward Error Coding (FEC) with very high processing cost per packet. The processing requirements of packets for each application have been calculated to approximate real applications on the test bed in (Pappu and Wolf, 2002). The capacity of the server is set to 2000Mcc/s (Million CPU cycles per second) and processing costs of IP Forwarding packets, Cast Encryption packets and FEC packets are uniformly distributed at 0.01Mcc, 0.116Mcc and 1.90Mcc, respectively. Simulations are carried out under three schedulers: SFQ, SWFQ, and WFQ.

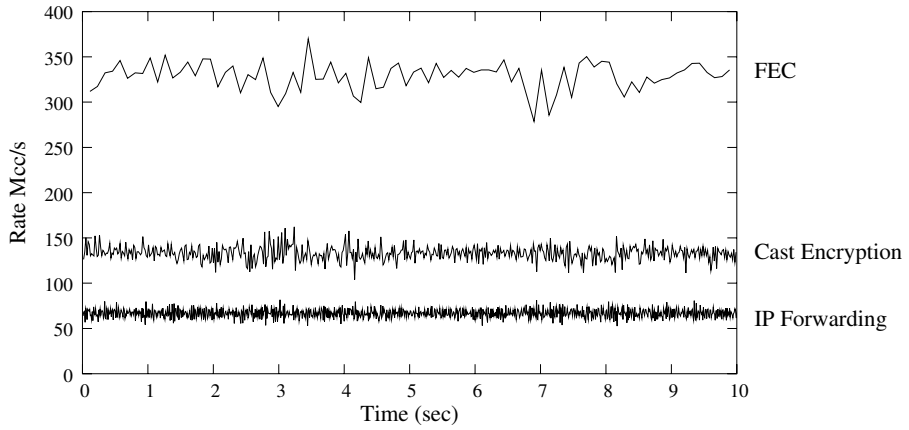
In the third experiment, we simulate another set of three applications: immersive audio mixing, MPEG2 Encoder and RC2 encryption. Processing requirements of audio mixing application was based on the traces of playing games in Section 8.2.1. Raw data from this traces are used this experiment. The traces of processing requirements of MPEG2 encoder application and RC2 encryption application are from (Sabrina and Jha, 2003).

8.5.2 Fairness of SWFQ

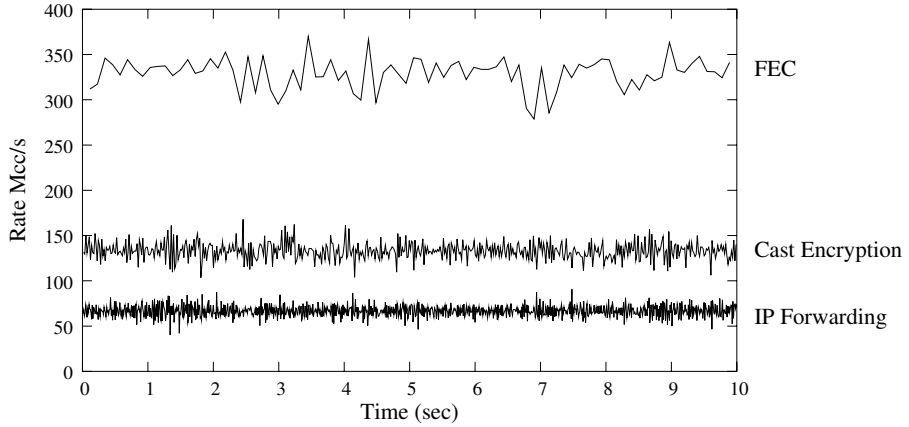
To evaluate the fairness of SWFQ, in the first simulation, we simulate ten IP Forwarding flows, five Cast Encryption flows, and two Forward Error Coding flows with the respective weights for each application flow being 1, 2, and 5. Each flow sends packets at random time interval and the average time interval is set based on the reserved rate. The processing rates are measured by averaging over a small time window which is set about 20 times the average time interval in each application flow. As shown in Fig. 8.7, the rate plots of the three applications under these schedulers are similar in small time intervals, therefore, it is demonstrated that SFQ and SWFQ have comparable fairness and throughput guarantees to WFQ.

8.5.3 Delay Properties of SWFQ

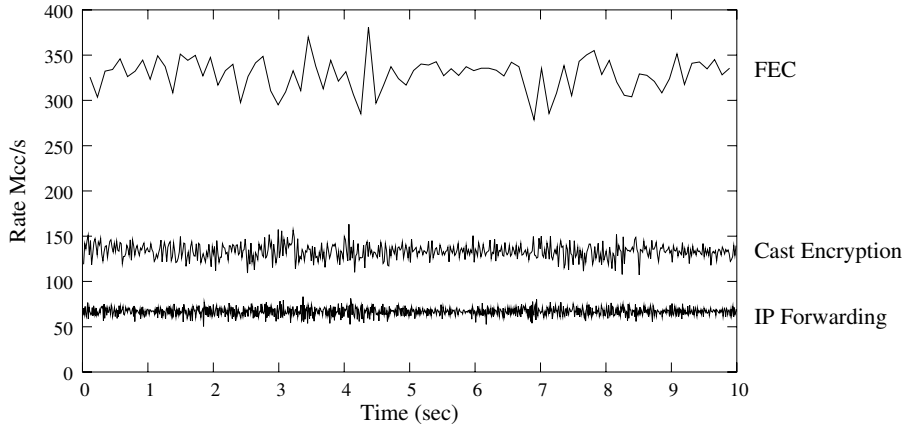
To compare the delay properties of SWFQ with SFQ and WFQ, in the second simulation, we simulate a total of 30 flows for IP Forwarding, Cast Encryption and Forward Error Coding, ten flows for each application. Each flow reserves the same processing rate, and sends packets randomly at specified average time intervals to just saturate its share. The sum of average processing rates of all flows is just under the server



(a) SFQ



(b) SWFQ



(c) WFQ

Figure 8.7 Processing rates allocated to IP Forwading, Cast Encryption, and FEC.

Schedulers	IP Forwarding	Cast Encryption	Forward Error Coding
SFQ	1.24	1.27	1.54
SWFQ	0.75	1.08	1.13

Table 8.1
Delay standard deviations of SWFQ and SFQ with IP Forwarding, Cast Encryption and Solomon Forward Error Coding (msec).

Parameters	Audio Mixing	RC2 Encryption	MPEG2 Encoding
Processing Times	1.7 - 5.5 ms	1 - 3 ms	18 - 35 ms
Weights	1	1	3

Table 8.2
Processing requirements of Audio Mixing, RC2 Encryption and MPEG2 Encoding.

capacity. Therefore, the delays measured are mainly due to scheduling not due to queueing backlog. The results in Fig. 8.8 and Fig. 8.9 show that SWFQ provides lower maximum delays to all application flows, especially, to FEC application flows, when compared with SFQ. As illustrated in Table 8.1, SWFQ also gives smaller delay standard deviations than SFQ for all applications flows, especially, for IP Forwarding and FEC flows. Reduction in delay standard deviations would reduce the delay jitters.

We expect that SWFQ would give better delay behavior due to its more accurate system virtual time, especially, where variations in processing requirements of packets are large (IP forwarding and FEC packets). During the processing time of a FEC packet, since the system virtual time in SFQ remains constant, packets arriving at empty session queues during this time can have start times equal to this FEC packet, and thus, are transmitted before packets waiting in backlogged sessions. As a result, packets waiting in backlogged sessions experience larger delays. Fig. 8.8 shows that WFQ provide smaller maximum delays than SWFQ and SFQ for application flows that have low processing time per packet to reserved rate ratio. However, SWFQ can provide lower maximum delays for FEC packets when compared with WFQ. This behavior was mentioned in the discussion on *delay guarantees* earlier.

In the third simulation, we simulate 30 applications including ten audio mixing ap-

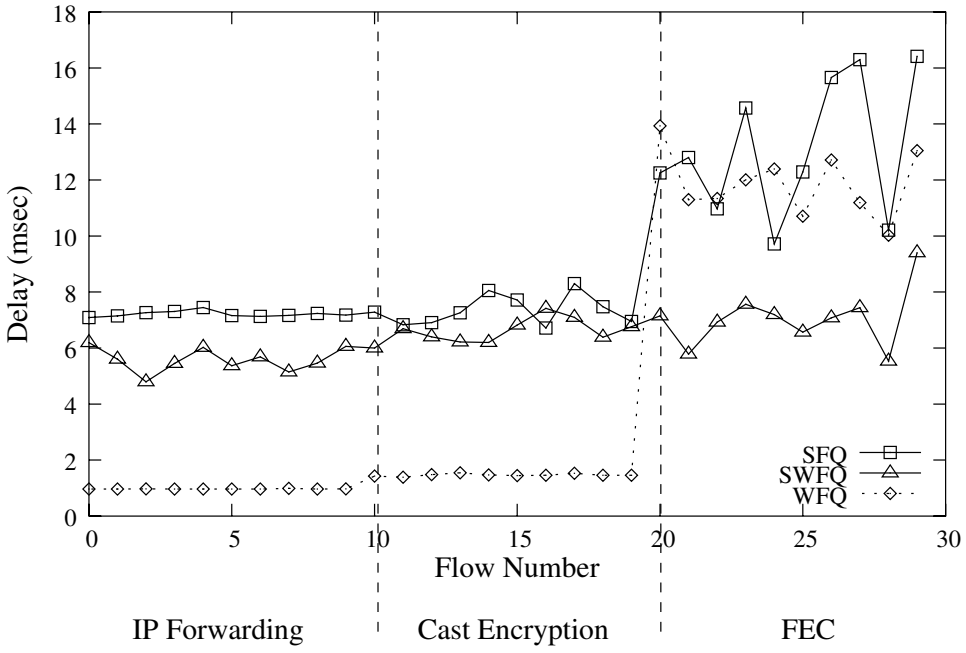
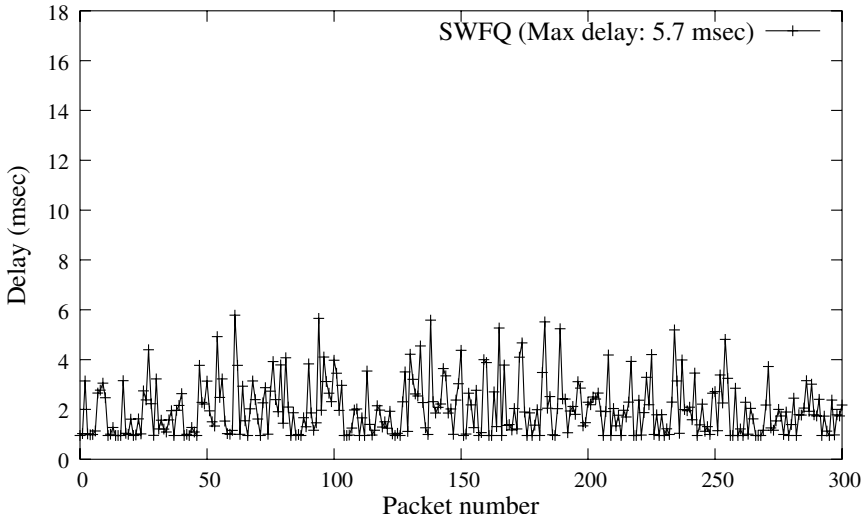
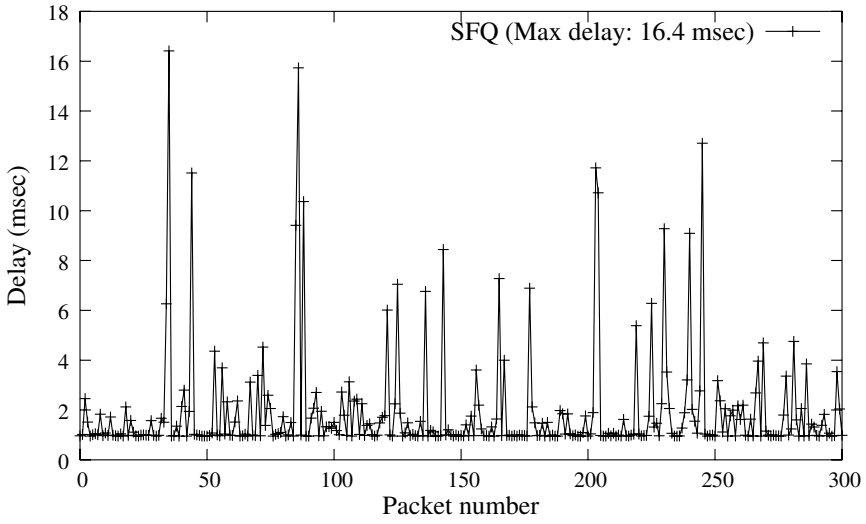


Figure 8.8 Maximum delays of packets in IP Forwarding (queues: 0-9), Cast Encryption (queues: 10-19), and FEC (queues:20-29).



(a) SWFQ



(b) SFQ

Figure 8.9 Delay performance of SWFQ and SFQ with Forward Error Coding application.

Schedulers	Audio Mixing	RC2 Encryption	MPEG2 Encoding
SFQ	6.88	7.07	5.72
SWFQ	5.36	5.83	4.13

Table 8.3
Delay standard deviations of SWFQ and SFQ with Audio Mixing, RC2 Encryption and MPEG2 Encoding (msec).

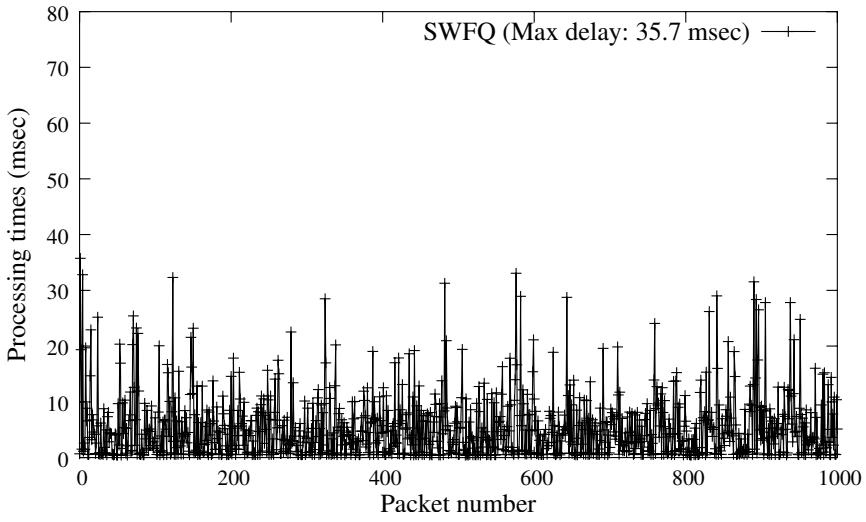
plications, ten RC2 encryption application flows and ten MPEG2 Encoder application flows. The processing requirements of each application and reserved processing weights at the server are shown in Table 8.2. In each applications, packets arrive at the server randomly with specified average time interval. The sum of average processing requirements of these applications are just under server capacity so that the delays measured are mainly due to scheduling not queueing backlog.

As shown in Figures 8.10 - 8.12, SWFQ also provides smaller maximum delays than SFQ in all three applications. In particular, SWFQ can reduce the maximum delays of audio mixing, RC2 encryption and MPEG2 encoding by 42%, 33% and 59%, respectively. Table 8.3 also shows the reduction in delay standard deviation of SWFQ compared with SFQ.

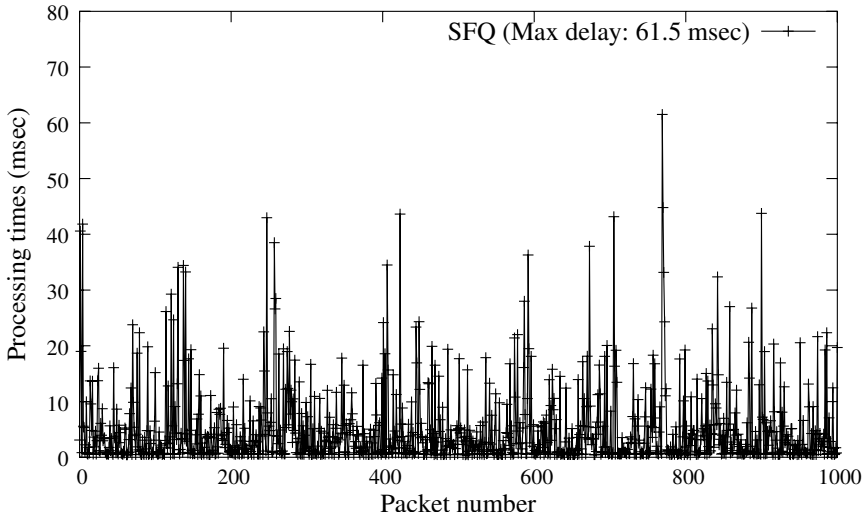
Another simulation study in (Sabrina and Nguyen, 2005) on three applications, namely MPEG2, RC2 encryption and RC2 decryption, also shows the advantage of SWFQ over SFQ in reducing the worst case delay. In particular, by using SWFQ, the worst case delay can be reduced by 23% for MPEG2 flow, 72% for RC2 decryption data flow and 60% for RC2 encryption data flow compared with the delays achieved when using SFQ.

8.6 Conclusions

In this chapter, we present a processing resource management architecture for supporting the immersive audio mixing application as well as other real-time content creation applications. We first briefly describe processing requirements of an immersive audio mixing server. Then, a classification of applications with processing re-

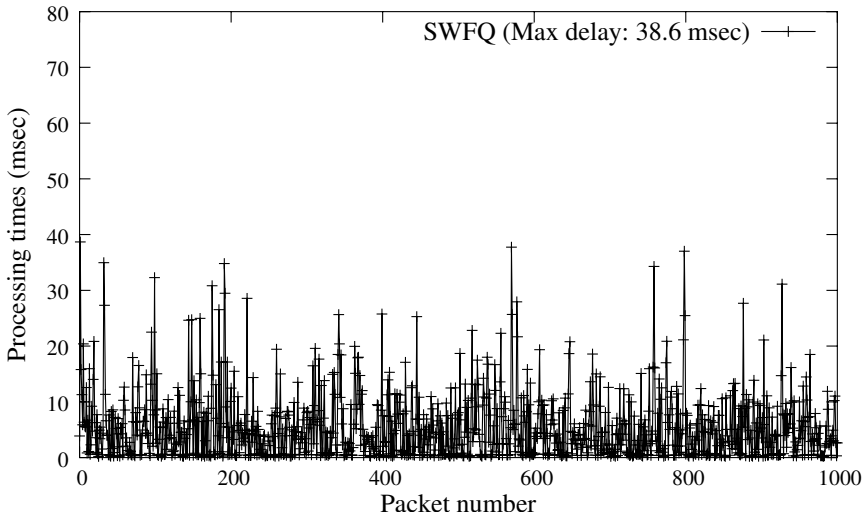


(a) SWFQ

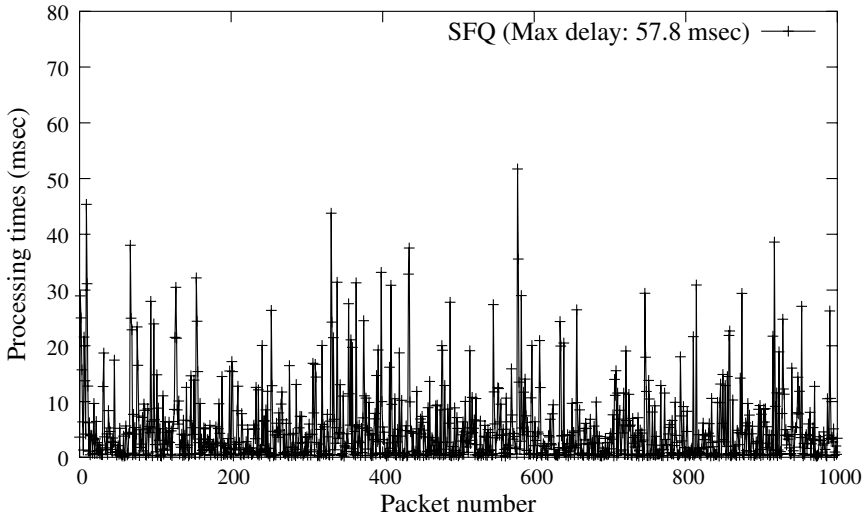


(b) SFQ

Figure 8.10 Delay performance of SWFQ and SFQ with Audio mixing applications.

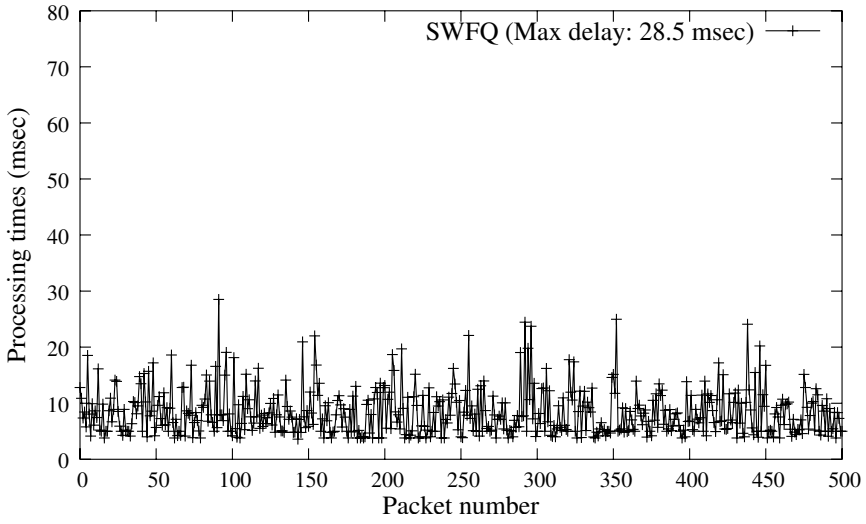


(a) SWFQ

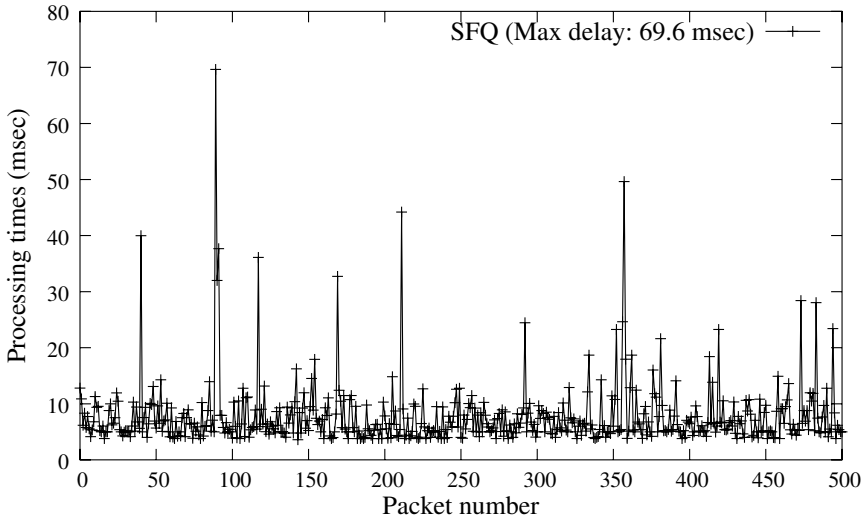


(b) SFQ

Figure 8.11 Delay performance of SWFQ and SFQ with RC2 Encryption application.



(a) SWFQ



(b) SFQ

Figure 8.12 Delay performance of SWFQ and SFQ with MPEG2 Encoding application.

source requirements and QoS provisioning issues are discussed. Due to the inability of determining processing times for scheduling, a processing resources scheduling algorithm called Start-time Weighted Fair Queueing is proposed.

This chapter also describes how SWFQ can be used in accordance with SWON to improve the cost of delivery and latency of the immersive voice communication service. In particular, we assume that a SWON switch is connected to a high-performance processing server which could be running a particular operating system. SWFQ can be implemented at the switch, which schedules packets from different real-time application flows and deliver these to the server for processing. Since end-to-end

From analysis and simulation, it is shown that SWFQ offers good fairness and delay properties. In fact, the fairness of SWFQ is comparable to WFQ and the delay behavior is better than SFQ. In particular, simulations on various applications shows that SWFQ can reduce the worst case delay up to 60% compared with SFQ. We propose to use SWFQ for processing resources scheduling to support QoS in two categories: processing resource reservation and best-effort.

Chapter 9

Conclusions

9.1 Overview

In the current Internet, various emerging applications require real-time creation and distribution of content. This thesis concentrates on the provision of an immersive voice communication service to massive multiplayer online games, which requires real-time creation of audio scenes and large-scale distribution of audio streams. We present several delivery architectures for this service and provide suitable server assignment algorithms for each architecture. This thesis evaluates the performance of these architectures in different game delivery scenarios. In addition, we develop a suitable server processing resource management for sharing processing resources among various real-time content creation applications including the immersive audio mixing application. In this chapter, we provide a summary of the main contributions and findings throughout thesis, recommendations from the thesis, and future work.

9.2 Summary of Contributions and Findings

The main contribution of the thesis is the investigation of different delivery architectures for providing an immersive voice communication service to multiplayer online games. We develop suitable server assignment algorithms of each architecture and evaluate the performance of delivery architectures using these algorithms. Another contribution of the thesis is the development of a processing resource scheduling

algorithm for managing resource sharing between the immersive audio mixing application and other real-time content creation applications. This section describes main contributions throughout the thesis.

9.2.1 Classification of Architectures and Performance Evaluation Framework

The thesis evaluates different delivery architectures for the provision of immersive voice communications to multiplayer online games based on existing game server architectures. These architectures include peer-to-peer, central server, distributed locale servers, and distributed proxies. We describes the main characteristics, and advantages and limitations of these delivery architectures.

In addition, this thesis develops a game simulation model for massively multiplayer online games. The model consists of player distribution in the Internet and different avatar distribution in the virtual world. We also introduce key parameters for the performance evaluation of delivery architectures. This game simulation model does not appear to be available in the literature.

9.2.2 Central Server

The central server architecture is used in most current massively multiplayer online games. The contribution of this thesis is to investigate the best strategy when using a central server for creating immersive audio scenes for the game. We assume that a game service provider can have access to a multitude of potential processing sites and propose two optimization objectives for choosing an optimal central server from this set. This thesis also proposes a dynamic relocation of a central server in response to changes in player distribution due to time zone differences. The simulation results show that relocation of the central server in response to these changes can significantly reduce the interactive delay by up to 40% and the network bandwidth usage by up to 50%. In addition, the optimal central server can reduce the interactive delay of a randomly located central server by up to 50%.

9.2.3 Distributed Locale Servers

In the distributed locale server, we develop an Integer Linear Programming (ILP) model and the greedy heuristics for solving the server assignment problem. These algorithms are developed for optimizing communication delay in the game while related research efforts in the literature mainly consider load balancing and load sharing among a distributed set of servers. This is a major contribution of this thesis.

A significant contribution in this research is the development of a new multi-layer graph model for efficiently solving the ILP problem by transforming the problem to a graph problem. A greedy heuristics is developed by using this graph and has significantly lower computation complexity compared to the ILP model. In addition, simulation results show that the greedy heuristic solutions are within 5% of the optimal in all cases. As it will be shown later, this multi-layer graph is also applicable to the proxy assignment problem in the distributed proxy architecture. We believe that this multi-layer graph approach can be applicable to other problems in the literature.

By using these proposed server assignment algorithms, the distributed locale server architecture improves the performance of the central server significantly. In particular, it is shown that increasing the number of servers reduces the latency of the optimal central server significantly when there is a physical/virtual world correlation. With a reasonable number of servers, the use of distributed locale servers can reduce by up to 60% (average of 20%) the delay of the central server. In addition, distributed locale servers outperform the central server in both latency and network bandwidth usage, especially when there is virtual world/physical network correlation.

9.2.4 Distributed Proxy Architecture

In the distributed proxy architecture, a key contribution of this thesis is the development of proxy assignment algorithms. While several research projects in the literature have proposed a proxy-based system for networked games, the provision of these systems based on the game characteristics including player distribution and avatar distribution is not adequately addressed. There are several related research studies in proxy location problems, however, these are mainly designed for web con-

tent and can not be applied for the distributed proxy architecture in this thesis.

We first develop an ILP model for optimal proxy assignment. This ILP model has the same mathematical characteristics as the ILP for the distributed locale servers but is different in the way decision variables are represented. We again adapt the multi-layer graph approach and devise a greedy heuristics for solving the problem. The results show that the ILP model is very unscalable while the greedy heuristics is highly efficient. The greedy heuristics is also significantly more accurate than a common approach of assigning a player to the closest proxy.

Another contribution in the distributed proxy architecture is the evaluation of the bandwidth saving of network multicast. The simulation experiments show that multicast is effective only when the average number of avatars in hearing range is more than 20 (i.e large clan or crowd). In this situation, multicast can reduce the network bandwidth of unicast by 50%. In these scenarios, when game players are connected to a small number of ISP POPs, multicast is more effective. However, when game players are widely spread, the efficiency of multicast is reduced significantly.

9.2.5 Performance Comparison Evaluation

Another major contribution of this thesis is the performance evaluation of all delivery architectures. Extensive simulation experiments have been carried out. This quantitative study will be of benefit to future immersive voice service providers in the design of a cost effective delivery architecture for this service. From this study, we provide recommendations on choosing suitable delivery architectures based on the server resource availability, multicast, and game's avatar aggregation behaviors. These recommendations are summarized in the next section.

9.2.6 Server Resource Management

The last contribution of the thesis is the design of a resource management architecture for sharing processing resources among various real-time content creation applications such as media transcoding, encryption, and the immersive audio mixing application. We propose a processing resource scheduling algorithm called Start-time

Weighted Fair Queueing (SWFQ). From analysis and simulation, it was shown that SWFQ offers good fairness and delay properties compared with current schemes. In fact, the fairness of SWFQ was comparable to Weighted Fair Queueing (WFQ) and the delay behavior is better than Start-time Fair Queueing (SFQ). In particular, SWFQ can reduce the worst case delay of SFQ by up to 60%.

9.3 Thesis Recommendations

9.3.1 Recommendations on Infrastructure Support

As discussed in Chapter 2, it would be costly for each immersive voice service provider to deploy its own hardware infrastructure. We envisage that a shared server infrastructure is suitable for providing the network and server infrastructure for various immersive voice service providers (i.e, those providing this service for different games). One of shared server infrastructures of interest is the SWON architecture (Boustead et al., 2004). SWON is particularly suitable for those that provide the immersive voice service to MMOG. Two key characteristics of MMOG are: 1 - The number of participants ranges from several hundreds to more than ten thousands and those numbers tend to increase significantly when the games become popular; 2 - Participants are from widely diverse geographical locations. These characteristics can be effectively addressed by SWON.

In addition to the economic benefit of using shared infrastructure, two key advantages of using SWON are:

- *Scalability and dynamic adjustment:* An immersive voice service provider for a particular game can hire virtual servers from SWON's server farms. The capacity of these virtual servers and the number of virtual servers can be dynamically adjust based on the game requirement.
- *Latency driven distribution:* The use of distributed virtual servers from SWON's server farms located in large geographic span are particularly suitable for the distributed locale server and distributed proxy architecture.

In terms of server assignments, both optimization objectives provided in Chapter 4 are suitable for the central server architecture. Relocation of a central server between two SWON servers is also possible. In distributed locale server and distributed proxy architectures, both greedy heuristics are suitable for practical implementation.

In each SWON server site, SWFQ can be implemented for supporting various real-time content creation applications in two QoS categories: processing resource reservation and best-effort. Depending on the number of players assigned to a SWON server, a game service provider can reserve a certain amount of processing resource.

9.3.2 Recommendations on Delivery Architectures

From the work performed in this thesis, we propose the following recommendations.

- *Avatar Aggregation Behaviors*

  For loner based games, the peer-to-peer architecture would be favored but security need to be addressed. For clan/crowd based games, both distributed server architectures are suitable, especially, when there is a correlation between the virtual world and the physical world. In this case, the distributed proxy architecture provides lower delay than the distributed locale server architecture but requires larger amount of network bandwidth. A hybrid architecture may be considered in scenarios in which, parts of the virtual world that mainly consist of loners would use the peer-to-peer architecture, while crowds or clans would use one of these distributed server architectures.

- *Bandwidth Resource Availability*

  In addition to avatar aggregation behaviours, resource availability should be taken into consideration. While loner based games should use peer-to-peer, in clan/crowd based games, distributed locale servers and the central server are more suitable when the network bandwidth is scarce. When the core bandwidth is not a constraint, distributed proxies are more suitable.

- *Server Availability*

When only a small number of servers is available, central server and distributed locale servers are more suitable than distributed proxies. When a large number of server is available, distributed proxies are more suitable due to lower latency performance.

- *Physical/Virtual World Correlation*

  When there is physical/virtual world correlations, distributed server architectures and peer-to-peer are more suitable than the central server architecture.

- *Multicast*

  With respect to the distributed proxy architecture, multicast is only effective when the number of avatars in hearing range is large. In addition, if game players are connected to a small number of ISP POPs, the bandwidth saving of multicast is more significant. When game players are widely spread, the efficiency of multicast is reduced considerably.

9.4 Future Work

The work in this thesis provides a clear path for future work. This section presents opening research issues for further investigation.

9.4.1 Performance Evaluation Model

Mobility

We have not considered movements of avatars in our analysis. In distributed locale servers, due to avatar movements, the current assignment of locales to server can drift from optimal. As time goes by, it is necessary to rerun server assignment algorithms to re-assign locales to servers. An interesting future work is to introduce avatar movement patterns and investigate how often we need to reassign servers and portions of audio streams need to be rerouted.

Resources Constraints

While latency is the key priority in interactive entertainment applications, deployments need to cope with the availability of resources including server processing capacity and bandwidth. Since our model of each delivery architecture focuses on latency optimization a future extension of this work is to add server resource constraints into the model. For the central server and distributed proxies, it is easy to add into the model by doing some simple checks. The extension of server assignment algorithms for distributed locale servers and distributed proxies, which consider resource constraints, has been also outlined in Section 5.2.3 and Section 6.2.3, respectively.

In addition, due to limited server capacity, server processing delay may need to be considered. This delay can vary depending on each server load and utilization. Our proposed optimization algorithms do not consider this delay. For the distributed locale server and distributed proxy architectures, designing new optimization algorithms that takes into account these delays will be an interesting future work.

Multiple Game Service Provisioning Model

In practice, the server infrastructure may need to provision the immersive voice communication service to several games simultaneously. In this scenario, it is necessary to develop new provisioning algorithms that take into account server resource limits, player distributions and avatar distributions of all these games. Each game service provider can decide to hire a virtual central server or distributed virtual servers depending on the game's requirement and player distribution. It is up to the server providers to assign servers to these games in a way that satisfies the performances of these games and also optimizes the server provider's resource usages. The provisions can be done in parallel (where the service is initiated for all these games at the same time) or in sequence depending on each game's service request time. Developing mechanisms and optimization algorithms for these provisions will be an interesting area of work.

Evaluation based on Real Physical/Virtual World

It is an interesting future work to evaluate the performance of delivery architectures

based on the current Internet topology and actual avatar positions and player distribution obtained from a current MMOG. Mapnet [1] can be used to create the Internet backbone consisting of link bandwidth and delay of multiple international backbone providers. Avatar positions and players' IP addresses can be obtained from a current MMOG game server. Players' IP addresses can be translated to geographical locations by using some software such as IP Locator [2]. The combination of player geographical locations and the Internet topology obtained from Mapnet will give a realistic model of the game physical world.

9.4.2 Experimental Investigation

The future work is to build a test bed to implement these architectures in the Internet. The test bed will consist of a number of server located in different geographical regions. Implementations of these architectures for providing immersive voice communications to a current network game will be an interesting future work. This will enable us to have Internet wide trials and obtain experimental delay measurements and user perception of voice quality.

Another interesting experimental work is to implement SWFQ in a server and investigate its use for sharing processing resources among several immersive voice mixing applications and other applications such transcoding and encryption. Experimental delay and fairness measurements from this would prove practical advantages of this algorithm.

[1] Mapnet, http://www.caida.org/tools/visualization/mapnet/
[2] IP Locator, http://www.geobytes.com/IpLocator.htm

Bibliography

Aas, T. A., Brown, S., Green, B., and Motte, S. (2003). Show me the money! Revenue models for massively multiplayer games. Game developers conference. available at www.gdconf.com/archives/2003/.

Agarwal, G., Chah, R., and Walrand, J. (2001). Content distribution using network layer anycast. In *Proceedings of Second IEEE Workshop on Internet Applications*, pages 124–132.

Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R. (2001). Resilient overlay networks. In *Proc. 18th ACM Symposium on Operating Systems Principles*, pages 131–145.

Anderson, D. B., Barrus, J. W., Howard, J. H., Rich, C., Shen, C., and Waters, R. C. (1995). Building multi-user interactive multimedia environments at MERL. *IEEE Multimedia*, 2(4):77–82.

Armitage, G. (2001). Sensitivity of Quake3 player to network latency. SIGCOMM Internet Measurement Workshop. Poster.

Banerjee, S., Bhattacharjee, B., and Kommareddy, C. (2002). Scalable Application Layer Multicast. In *Proc. SIGCOMM*.

Banga, G. and Druschel, P. (1999). Resource containers: a new facility for resource management in server systems. In *Proc. of the 3rd Symposium on Operating Systems Design and Implementation*. ACM.

Barish, G. and Obraczka, K. (2000). World wide web caching: trends and techniques. *IEEE Communication Magazine*.

Barrus, D. B., Waters, R., and Anderson, D. B. (1996). Locales: Supporting large multiuser virtual environments. *Computer Graphics and Applications, IEEE*, 16(6):50–57.

Bauer, D., Rooney, S., and Scotton, P. (2002). Network infrastructure for massively distributed games. In *Proceedings of the first workshop on network and system support for games*, pages 36–43. ACM Press.

Bennett, J. C. R. and Zhang, H. (1997). Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689.

Bernier, Y. (2001). Latency compensating methods in client/server in-game protocol design and optimization. In *2001 Game Developers Conference*, San Francisco, California.

Bolot, J. C. and Parisis, S. F. (1998). Adding voice to distributed games on the Internet. In *the Proceedings of Seventeenth Annual Joint Conference of the IEEE Computer and Communication Societies*, pages 480–487.

Boustead, P., Safaei, F., and Dowlatshahi, M. (2005). DICE: Internet delivery of immersive voice communication for crowded virtual spaces. In *Proceeding of IEEE International Conference on Virtual Reality*, pages 35–41, Bonn, Germany. IEEE.

Boustead, P., Safaei, F., and Nguyen, V. (2004). Switched overlay networks (SWON): switching support for a global network of virtual servers. Technical report, Smart Internet Technology CRC, http://www.titr.uow.edu.au/ paul/SWON1.pdf.

Bovy, C., Mertodimedjo, H., Hooghiemstra, G., Uijterwaal, H., and Mieghem, P. (2002). Analysis of end-to-end delay measurements in internet. Technical report, RIPE, http://www.ripe.net/test-traffic.

Bryhni, H., Klovning, E., and Kure, O. (2000). A comparison of load balancing techniques for scalable web servers. *IEEE Network*, pages 58–64.

Calvert, K. L., Doar, M. B., and Zegura, E. W. (1997). Modeling Internet Topology. *IEEE Communication Magazine*, 35(6):160–163.

Campbell, A. T., De Meer, H. G., Kounavis, M. E., Miki, K., Vincente, J. B., and Villela, D. (1999). A survey of programmable networks. *Computer Communication Review*, 29(2):7–23.

Cardellini, V., Casalicchio, E., Colajanni, M., and Yu, P. S. (2002). The state of the art in locally distributed web-server systems. *ACM Computing Surveys*, 34(2):263–311.

Carlsson, C. and Hagsand, O. (1993). DIVE - a multiuser virtual reality system. In *IEEE Virtual Reality Annual Intl. Symp.*, pages 394–400. IEEE.

Carpenter, B. (2003). What is grid computing? Technical Report 11, Internet Society, http://www.isoc.org/briefings/011/. Grid Computing ISOC Member Briefing.

Carpenter, B. and Brim, S. (2002). Middleboxes: Taxonomy and Issues. Request for Comments 3234, IETF Network Working Group.

Casner, S. and Deering, S. (1992). The first IETF Internet audiocast. *ACM Computer Communication Review*, 22(3):92–97.

Chambers, C., Feng, W., and Feng, W. (2003). A geographic redirection service for on-line games. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 227 – 230. ACM.

Choi, S. and Shavitt, Y. (2003). Proxy location problems and their generalization. In *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)*, pages 898–904.

Choi, S. Y., Turner, J. S., and Wolf, T. (2003). Configuring Sessions in Programmable Networks. *Computer Networks*, 41:269–284.

Chu, Y., Rao, S. G., Seshan, S., and Zhang, H. (2001). Enabling Conferencing applications on the Internet using overlay multicast architecture. In *Proc. ACM SIGCOMM*. ACM Press.

Chu, Y., Rao, S. G., and Zhang, H. (2000). A case for end system multicast. In *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12. ACM.

Chuang, J. C. I. and Sirbu, M. A. (2001). Pricing multicast communication: a cost-based approach. *Telecommunication Systems*, 17(3):281–297.

Cooper, I., Melve, I., and Tomlinson, G. (2001). Internet web replication and caching taxonomy. Request for Comments 3040, IETF Network Working Group.

Cranor, C. D., Green, M., Kalmanek, C., Shur, D., Sibal, S., Van der Merwe, J. E., and Sreenan, C. J. (2001). Enhanced streaming services in a content distribution network. *IEEE Internet Computing*, pages 66–75.

Cronin, E., Filstrup, B., Kurc, A., and Jamin, S. (2002a). An Ecient Synchronization Mechanism for Mirrored Game Architectures. In *Proc. NetGames 2002*, pages 67–73. ACM.

Cronin, E., Filstrup, B., and Kurc, A. R. (2001). A distributed multiplayer game server system. Technical report, UM EECS589 Course Project Report, citeseer.ist.psu.edu/cronin01distributed.html.

Cronin, E., Jamin, S., Jin, C., Kurc, A. R., Raz, D., and Shavitt, Y. (2002b). Constrained mirror placement on the Internet. *IEEE Journal on Selected Areas in Communications*, 20(7).

Deering (1989). Host extensions for IP multicasting. Request for Comments 1112, IETF Network Working Group.

Demers, A., Keshav, S., and Shenker, S. (1990). Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1(1):3–26.

Diot, C. and Gautier, L. (July/August 1999). A distributed architecture for multiplayer interactive applications on the internet. *IEEE Networks magazine*, pages 6–15.

Douglis, F. and Kaashoek, M. (2001). Scalable Internet Services. *IEEE Internet Computing*, pages 36–37.

Duan, Z., Zhang, Z.-L., and Hou, Y. T. (2003). Service overlay networks: slas, qos, and bandwidth provisioning. *IEEE/ACM Transactions on Networking*, 11(6):870–883.

Elson, J. and Cerpa, A. (2003). Internet Content Adaptation Protocol (ICAP). Request for comments, IETF, RFC 3507.

Fall, K. and Varadhan, K. (2002). *The ns manual*. The VINT Project, http://www.isi.edu/nsnam/ns/.

Feng, W.-c. and Feng, W.-c. (2003). On the geographic distribution of on-line game servers and players. In *Proceedings of the second workshop on network and system support for games*. ACM Press.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. Request for Comments 2616, IETF Network Working Group.

Flak, T. and Stumm, M. (2002). Challenges In Providing Carrier-Grade Telephony Over Broadband Wireless Networks. *Internet Telephony*. http://www.tmcnet.com/it/0502/0502fso.htm.

Foster, I. and Kesselman, C. (2004). *The Grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers, second edition.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *Intl. Journal of High Performance Computing Applications*, 15(3):200–222.

Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. (2001). IDMaps: a global Internet host distance estimation service. *IEEE/ACM Transactions on Networking*, 9(5):525–540.

Fraser, K., Hand, S., Harris, T., Leslie, I., and Pratt, I. (2001). The XenoServer computing infrastructure. Technical report, Cambridge University.

Frecon, E., Greenhalgh, C., and Stenius, M. (1999). The DiveBone - an application-level network architecture for Internet-based CVEs. In *VRST*, pages 58–65. ACM.

Galtier, V., Mills, K. L., Carlinet, Y., Bush, S. F., and Kulkarni, A. B. (2001). Predicting and controlling resource usage in a heterogeneous active network. In

Proceedings of the Third International Workshop on Active Middleware Services, pages 35–44.

Gardner, B. and Martin, K. (1994). HRTF measurements of a KEMAR dummy-head microphone. Technical Report Technical Report 280, MIT Media Lab.

Gauthier, P., Cohen, J., Dunsmuir, M., and Perkins, C. (1999). Web Proxy Auto-Discovery Protocol. Technical report, IETF.

Gleeson, I., Lin, A., Heinanen, J., Armitage, G., and Malis, A. (2000). A framework for IP based virtual private networks. Request For Comment RFC 2764, IETF Network Working Group.

Golestani, S. J. (1994). A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM '94*, pages 636–646.

Goyal, P., Guo, X., and Vin, H. M. (1996). A hierarchical CPU scheduler for multimedia operating systems. In *Proc. of Operating System Design and Implementation*, pages 107–122.

Goyal, P., Lam, S., and Vin, H. (1995). Determining end-to-end delay bounds in heterogeneous networks. In *Proc. Workshop on Networks and Operating System Support for Digital Audio and Video*, pages 287–298.

Goyal, P. and Vin, H. M. (1997). Generalized Guaranteed Rate Scheduling Algorithms: A Framework. *IEEE/ACM Transactions on Networking*, 5(4):561–571.

Goyal, P., Vin, H. M., and Cheng, H. (1997). Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Transactions on Networking*, 5(5):690–704.

Green, M., Cain, B., Tomlison, G., and Thomas, S. (2000). CDN peering architectural overview. Technical report, IETF Network Working Group, http://quimby.gnus.org/internet-drafts/draft-green-cdnp-gen-arch-02.txt.

Greenberg, A. G. and Madras, N. (1992). How fair is fair queuing? *Journal of the Association for Computing Machinery*, 39(3):568–598.

Greenhalgh, C. and Benford, S. (1999). Supporting rich and dynamic communication in large-scale collaborative virtual environments. *Presence: Teleoperators and Virtual Environments*, 8(1):14–35. MIT Press.

Harrington, J. and Cassidy, S. (1999). *Techniques in Speech Acoustics*. Kluwer.

Hendrix, C. and Barfield, W. (1996). Presence within virtual environments as a function of visual display parameters. *Presence*, 5(3):274–289. MIT Press.

Hew, K., Gibbs, M. R., and Wadley, G. (2004). Usability and sociability of Xbox Live voice channel. In *Proc. of Australian Workshop on Interactive Entertainment (IE2004)*, pages 51–58.

Kyriakakis, C. (1998). Fundamental and technological limitations of immersive audio systems. *Proceedings of the IEEE*, 86(5):941–951.

Kyriakakis, C., Tsakalides, P., and Holman, T. (1999). Surrounded by sound. *IEEE Signal processing magazine*, 16(1):55–66.

Lui, J. and Chan, M. (2002). An efficient partitioning algorithm for Distributed Virtual Environment Systems. *IEEE Transactions on Parallel and Ditributed Systems*, 13(3):193–211.

Lui, J. C. S., So, O. K. Y., Chan, M. F., and Tam, T. S. (1998). Dynamic partitioning for a Distributed Virtual Environment. In *Proceedings of the 3rd High Performance Computing Asia Conference (HPC Asia'98)*.

Ma, W. Y., Shen, B., and Brassil, J. (2001). Content Services Network: the architecture and protocols. In *the Proceedings of the 6th International Workshop on Web Caching and Content Distribution*, pages 83–101, Boston, Massachusetts.

Mahanti, A., Williamson, C., and Eager, D. (2000). Traffic analysis of a web proxy caching hierachy. *IEEE Network*, pages 16–23.

Mauve, M., Fischer, S., and Widmer, J. (2002). A generic proxy system for networked computer games. In *Proceedings of the first workshop on network and system support for games*, pages 25–28. ACM Press.

Mouchtaris, A., Reveliotis, P., and Kyriakakis, C. (2000). Inverse filter design for immersive audio rendering over loudspeakers. *IEEE Transactions on Multimedia*, 2(2):77–87.

Moy, J. (1998). OSPF version 2. Request for Comments 2328, IETF Network Working Group.

Nguyen, C. D., Platt, D., and Safaei, F. (2003a). Design of processing resources scheduling in programmable networks. In *Proceedings of Australian Telecommunications Network and Applications Conference (ATNAC)*, Melbourne. ISBN: 0-646-42229-4.

Nguyen, C. D., Safaei, F., and Boustead, P. (2004a). Comparison of distributed server architectures in providing immersive audio communication to massively multiplayer online games. In *Proceedings of Australian Telecommunications Network and Applications Conference (ATNAC)*, pages 499–505, Sydney. ISBN: 0-646-44190-6.

Nguyen, C. D., Safaei, F., and Boustead, P. (2004b). A distributed proxy system for provisioning immersive audio communication to massively multi-player games. In *Proceedings of ACM SIGCOMM Workshop on Network and Systems Support for Games (Netgames)*, page 166, Portland, Oregon, USA. ACM Press.

Nguyen, C. D., Safaei, F., and Boustead, P. (2004c). A distributed server architecture for providing immersive audio communication to massively multi-player online games. In *Proceedings of IEEE International Conference on Networks (ICON)*, pages 170–176, Singapre. IEEE.

Nguyen, C. D., Safaei, F., and Boustead, P. (2004d). Performance evaluation of a proxy system for providing immersive audio communication to massively multi-player games. In *Proceedings of 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), Globecom 2004*, pages 192–199, Dallas, TX, USA. IEEE.

Nguyen, C. D., Safaei, F., and Boustead, P. (2005a). Optimal assignment of distributed servers to virtual partitions for the provision of immersive voice com-

munication in massively multiplayer games. *To appear in Special Issue in Computer Communications Journal, Elsevier*.

Nguyen, C. D., Safaei, F., Boustead, P., and Brun, J. (2005b). An investigation of network and server architectures for the provision of immersive audio communications to massively multi-player online games. *Submitted to IEEE/ACM Transactions on Networking*.

Nguyen, C. D., Safaei, F., and Platt, D. (2004e). On the provision of immersive audio communication to massive multi-player online games. In *Proceeding of the Ninth IEEE Symposium on Computers and Communications*, pages 1000–1005, Alexandria, Egypt. IEEE Computer Society.

Nguyen, T. V., Chou, C. T., and Boustead, P. (2003b). Provisioning CDN over Shared Infrastructure. In *Proceedings of IEEE ICON*, pages 119–124, Sydney.

Pappu, P. and Wolf, T. (2002). Scheduling processing resources in programmable routers. In *Proceedings of IEEE INFOCOM 2002*, pages 104–112.

Parekh, A. K. and Gallager, R. G. (1993). A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357.

Peterson, L. (2001). NodeOS interface specification. Technical report, AN Node OS Working Group.

Peterson, L., Anderson, T., Culler, D., and Roscoe, T. (2002). A blueprint for introducing disruptive technology into the Internet. In *1st Workshop on Hot Topics in Networks (Hotnets-1)*, Princeton, New Jersey, USA.

Pulkki, V. (1999). Uniform spreading of amplitude panned virtual sources. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*.

Qie, X., Bavier, A., Peterson, L., and Karlin, S. (2001). Scheduling computations on a software-based router. In *Proc.IEEE Joint International Conference on Measurement & Modeling of Computer Systems (SIGMETRICS)*, Cambridge. IEEE.

Qiu, L., Padmanabhan, V. N., and Voelker, G. M. (2001). On the placement of web server replicas. In *IEEE INFOCOM*, pages 1587–1596.

Radenkovic, M. and Greenhalgh, C. (2002). Multi-party Distributed Audio Service with TCP Fairness. In *Proceedings of ACM Multimedia 2002*, pages 11–20. ACM.

Radenkovic, M., Greenhalgh, C., and Benford, S. (2002). Deployment issues for multi-user audio support in CVEs. In *ACM Symposium on Virtual Reality Software and Technology*, pages 179–185.

Reumann, J., Mehra, A., Shin, K. G., and Kandlur, D. (2000). Virtual service: a new abstraction for server consolidation. In *Proc. of the 2000 USENIX Annual Technical Conference*.

Rexford, J., Greenberg, A., and Bonomi, F. (1995). A fair leaky-bucket shaper for ATM networks. Technical report.

Rooney, S., Bauer, D., and Scotton, P. (2003). Efficient programmable middleboxes for scaling large distributed applications. In *Proc. IEEE OpenArch*, pages 65–74.

Sabrina, F. and Jha, S. (2003). Scheduling resources in programmable and active networks based on adaptive estimation. In *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks*, pages 2–11.

Sabrina, F. and Nguyen, C. D. (2005). et al., Processing resource scheduling in programmable nodes. *Elsevier Computer Communication Journal*, 28(6).

Safaei, F., Boustead, P., Nguyen, C. D., Brun, J., and Dowlatshahi, M. (2005). Latency driven distribution: infrastructure needs for participatory entertainment applications. *in IEEE Communication Magazine, Special Issue on Network Support for Interactive Entertainment Applications*, 43(5).

Safaei, F., Ouveysi, I., Zukerman, M., and Pattie, R. (March 2001). Carrier-scale programmable networks: wholesaler platform and resource optimization. *IEEE Journal on Selected Areas in Communications*, 19(3):566–573.

Savage, S., Collins, A., Hoffman, E., Snell, J., and Anderson, T. (1999). The end-to-end effects of Internet path selection. In *Proc. ACM SIGCOMM*, pages 289–299.

Schroeder, T., Goddard, S., and Ramamurthy, B. (2000). Scalable web server clustering technologies. *IEEE Network*, 14:38–45.

Sedgewick, R. (1990). *Algorithms in C*. Addison-Wesley.

Shaikh, A., Sahu, S., Rosu, M., Shea, M., and Saha, D. (2004). Implementation of a service platform for online games. In *SIGCOMM'04 Workshop on Network and System Support for Networked Games*, pages 106–110.

Singhal, S. and Zyda, M. (1999). *Networked virtual environments design and implementation*. ACM Press.

Stoica, I. (2004). Overlay networks. http://www.cs.virginia.edu/ cs757/slidespdf/757-09-overlay.pdf.

Subramanian, L., Stoica, I., Balakrishnan, H., and Katz, R. (2004). OverQoS: an overlay based architecture for enhancing internet QoS. In *Proc. of First Symposium on Networked Systems Design and Implementation (NSDI'04)*, pages 71–84. Usenix.

Ta, N. B. D. and Zhou, S. (2003). A dynamic load sharing algorithm for massively multiplayer online games. In *Proceedings of the IEEE International Conference on Networks*, pages 131–136.

Waters, R. C., Anderson, D. B., Barrus, J. W., Brogan, D., Casey, M., McKeown, S., Nitta, T., Sterns, I., and Yerazunis, W. (1997). Diamond park and Spline: a social virtual reality system with 3D animation, spoken interaction, and runtime modifiability. *Presence*, 6(4):461–481. MIT Press.

Waxman, B. (1988). Routing of multipoint connections. *IEEE Journal on Selected Areas on Communications*, 6(9):1617–1622.

West, D. B. (2001). *Introduction to graph theory*. Prentice Hall.

Whitaker, A., Shaw, M., and Gribble, S. (2002). Scale and performance in the delani isolation kernel. In *Proc. of the Fifth Symposium on Operating Systems Designs and Implementation*, Boston, MA.

Xie, G. G. and Lam, S. S. (1995). Delay guarantee of virtual clock server. *IEEE/ACM Transactions on Networking*, 3(6):683–689.

Zegura, E., Calvert, K., and Donahoo, M. (1997). A quantitative comparison of graph-based models for Internet topology. *IEEE/ACM Transactions in Networking*, 5(6):770–783.

Zou, L., Ammar, M., and Diot, C. (2001). An evaluation of grouping techniques for state dissemination in networked multi-user games. In *Proc. 9th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 33–40.

Appendix A

Proofs for SWFQ Analysis

A.1 Proof of Theorem 1

We use a similar method to the one in (Greenberg and Madras, 1992) to prove the result. Firstly, we prove the following lemma.

Lemma 1 *Suppose a packet just finished processing in SWFQ at time t. Then, the minimum start time of packets in current backlogged sessions at time t called S_j is always determined for scheduling and $S_j \leq V(t)$.*

This lemma can be proved by contradiction. In the GPS system, all backlogged sessions receive the same normalize service, which is equal to the system virtual time $V(t)$. In SWFQ, by serving packets in increasing order of start times, the system attempts to make the normalize service of each session equal to V(t). If all sessions in SWFQ have start times larger than $V(t)$, it means that all the sessions have been serviced more than it should have in the GPS server. Consequently, the work done in SWFQ will be higher than in GPS, which is impossible since both servers are work-conserving. When Lemma 1 is satisfied, the proof is carried out as follows.

Let k be the maximum number of packets in session i, whose service in GPS begins no later than time t. Since packet p_i^{k+1} has not started service in GPS at time t, we have $S_i^{k+1} \geq V(t)$. Let t_1 be the time SWFQ server begins to process packet p_i^{k+1}. From the result of Lemma 1, we have $V(t_1) \geq S_i^{k+1}$. As a result, we have

$V(t_1) \geq V(t)$. Since $V(t)$ is an increasing function, it follows that $t_1 \geq t$. Therefore,

$$W_i(0,t) \geq \sum_{j=1}^{k-1} P_i^j \geq \sum_{j=1}^{k} P_i^j - P^{max} \geq \hat{W}_i(0,t) - P^{max}$$

A.2 Proof of Theorem 2

We adapt the method of proof in (Rexford et al., 1995) for WFQ to prove the result for SWFQ. As shown in Theorem 1, if each of $(N-1)$ sessions in SWFQ leads GPS by P^{max}, then one session will lag GPS by $(N-1)P^{max}$. This bound is too large for many cases, we can prove a much smaller bound as follows.

It is noted that the throughput discrepancy is maximum when a packet starts service in SWFQ. Let t_i^k and $\hat{t}_i^k$ be the time packet p_i^k starts service in GPS and SWFQ, respectively. We only need to consider the case $t_i^k \leq \hat{t}_i^k$; otherwise, $W_i(0,\hat{t}_i^k) \leq \hat{W}_i(0,\hat{t}_i^k)$. Therefore,

$$\begin{aligned} W_i(0,\hat{t}_i^k) &= W_i(0,t_i^k) + W_i(t_i^k,\hat{t}_i^k) \\ &\leq \hat{W}_i(0,\hat{t}_i^k) + r_i(V(\hat{t}_i^k) - S_i^k) \end{aligned} \tag{A.1}$$

At time $\hat{t}_i^k$, since session i receives more service in GPS than in SWFQ, there must be a session j receiving more service in SWFQ than in GPS. Let p_j^m be the packet at the head of session j queue at time $\hat{t}_i^k$. As packet p_j^{m-1} must have finished service before packet k starts service in SWFQ, we have

$$S_j^{m-1} \leq S_i^k \tag{A.2}$$

Since packet p_j^m has not started at time $\hat{t}_i^k$ in SWFQ, and session j is receiving more service in SWFQ than in GPS, we also have

$$V(\hat{t}_i^k) \leq S_j^m \tag{A.3}$$

Applying these results into (7), we get,

$$\begin{aligned} W_i(0,\hat{t}_i^k) - \hat{W}_i(0,\hat{t}_i^k) &\leq r_i(S_j^m - S_j^{m-1}) \\ &\leq r_i \frac{P_j^{m-1}}{r_j} \\ &\leq r_i \max_{1 \leq n \leq N}(\frac{P_n}{r_n}) \end{aligned} \tag{A.4}$$

Combining the two cases, we have,

$$W_i(0,t) - \hat{W}_i(0,t) \leq min((N-1)P^{max}, r_i \max_{1 \leq n \leq N}(\frac{P_n}{r_n})) \tag{A.5}$$

A.3 Proof of Theorem 3

Let us denote d_i^k and $\hat{d}_i^k$ as the departure times of packet p_i^k in GPS and SWFQ, respectively. We have,

$$\begin{cases} d_i^k = t_i^k + \frac{P_i^k}{r_i \rho} \\ \hat{d}_i^k = \hat{t}_i^k + \frac{P_i^k}{r} \end{cases} \tag{A.6}$$

Where $\rho = \frac{r}{\sum_{i \in B(t)} r_i}$, and it represents the ratio of the service rate a session currently receiving to the service rate of that session when all sessions are backlogged in the GPS system.

$$\hat{d}_i^k - d_i^k = (\hat{t}_i^k - t_i^k) + (\frac{P_i^k}{r} - \frac{P_i^k}{r_i \rho}) \tag{A.7}$$

We only need to consider the case $t_i^k \leq \hat{t}_i^k$; otherwise, $\hat{d}_i^k \leq d_i^k$. Since $W_i(0, t_i^k) = \hat{W}_i(0, \hat{t}_i^k)$, we have,

$$\hat{W}_i(t_i^k, \hat{t_i^k}) = W_i(0, t_i^k) - \hat{W}_i(0, t_i^k) \tag{A.8}$$

Combining (14) with the result of Theorem 2, we have,

$$\hat{W}_i(t_i^k, \hat{t}_i^k) \leq \sum_{n \in B(t), n \neq i} (P_n^{max}) \tag{A.9}$$

The worst case delay occurs when the service of session i under SWFQ lag GPS by the maximum amount. As discussed in the proof of Theorem 2, it occurs when all other backlogged sessions in SWFQ lead GPS, only session i in SWFQ lags GPS. Therefore, in interval $(t_i^k, \hat{t}_i^k)$, the normalize service of session i is lower than all other backlogged sessions in SWFQ. Therefore, session i must receive service continuously in this interval. Consequently,

$$\hat{t}_i^k - t_i^k \leq \sum_{n \in B(t), n \neq i} \frac{P_n^{max}}{r} \tag{A.10}$$

The delay guarantee for GPS, given in (Goyal and Vin, 1997), is,

$$d_i^k = L_{GPS}(p_i^k) \leq EAT(p_i^k) + \frac{P_i^k}{r_i \rho} \tag{A.11}$$

Combining (13) (16) and (17), we have,

$$L_{SWFQ}(p_i^k) = \hat{d_i^k} \leq EAT(p_i^k) + \sum_{n \in B(t), n \neq i} \frac{P_n^{max}}{r} + \frac{P_i^k}{r} \tag{A.12}$$

A.4 Proof of Theorem 4

To derive the delay bound, we need to derive the bound between $EAT(p_i^k)$ and $A(p_i^k)$.

Let S_i be the set of packet indexes in flow i that has the follow property.

$$S_i = \{n | n > 0 \cap (EAT(p_i^{n-1}) + \tfrac{P_i^{n-1}}{r_i}) \leq A(p_i^n)\}$$

For each packet p_i^k, $\exists j : j \leq k$ that is the largest integer belonging to S_i. From the property of S_i above and the definition of *Expected Arrival Time*, we have,

$$EAT(p_i^k) = A(p_i^j) + \sum_{n=0}^{n=k-j-1} \frac{P_i^{j+n}}{r_i} \tag{A.13}$$

The sum in the above equation is the actual processing requirements of those packets indexing from j to k. Therefore,

$$EAT(p_i^k) = A(p_i^j) + \frac{AP(A(p_i^j), A(p_i^{k-1}))}{r_i} \tag{A.14}$$

From the definition of AP function, we have

$$AP(A(p_i^j), A(p_i^k)) \leq \delta_i + r_i(A(p_i^k) - A(p_i^j)) \tag{A.15}$$

As a result,

$$AP(A(p_i^j), A(p_i^{k-1})) \leq \delta_i + r_i(A(p_i^k) - A(p_i^j)) - P_i^k \tag{A.16}$$

From Equation A.14 and A.16 we have

$$EAT(p_i^k) \leq A(p_i^j) + \frac{\delta i}{r_i} + A(p_i^k) - A(p_i^j) - \frac{P_i^k}{r_i} \tag{A.17}$$

Therefore,

$$EAT(p_i^k) - A(p_i^k) \leq \frac{\delta i}{r_i} - \frac{P_i^k}{r_i} \tag{A.18}$$

Combining the above equation and the delay guarantee $EAT(p_i^k)$ derived in Theorem 3, the delay bound is

$$L_{SWFQ}(p_i^k) - A(p_i^k) \leq \frac{\delta_i}{r_i} - \frac{p_i^k}{r_i} + \sum_{n \in B(t), n \neq i} \frac{P_n^{max}}{r} + \frac{P_i^{max}}{r} \tag{A.19}$$

Appendix B

Details of Simulation Environments

B.1 Network Topologies

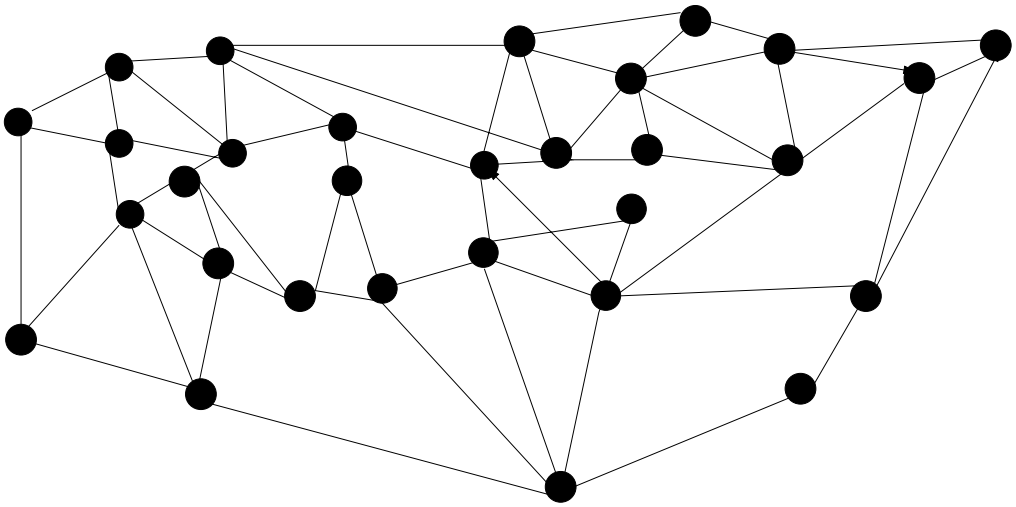
The network topologies used in simulations are generated by using GT-ITM topology generator package (Calvert et al., 1997). There are three major graph models included in this package: random graph, hierarchical graph, and transit-stub graph. Basically, these models are different in the ways nodes and edges are generated and distributed on a two-dimension plane.

In a random graph, edges are probabilistically added to a given set of nodes with no structure among nodes. There are several favours of random graphs including: pure random, Waxman model (Waxman, 1988), and other variations to the Waxman model. In a pure random graph, edges are probabilistically added to nodes with a fixed probability *p*. In the Waxman model, an edge from node *u* to node *v* are added with the following probability.

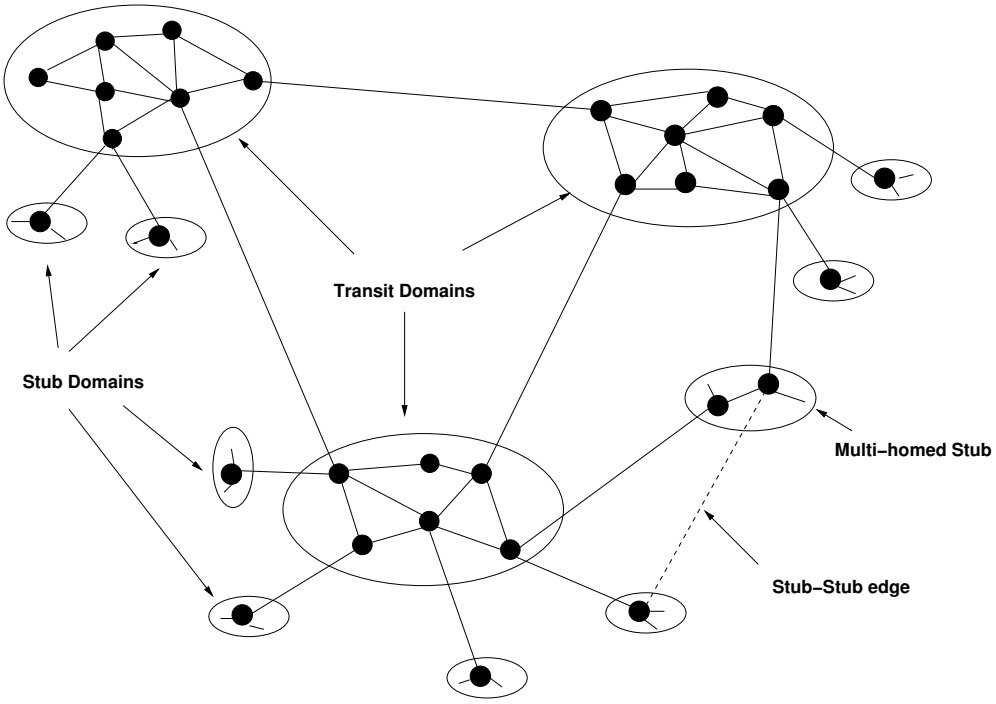
$$P(u, v) = \alpha e^{-d/(\beta L)} \tag{B.1}$$

This model is more realistic for modelling a real network compared with the pure random graph model. An example of a random graph is illustrated in Figure B.1b.

The above random graphs are also referred to as a *flat* graph model since there is no structure among the nodes. This does not closely capture real internetworks, in which some nodes are more likely to be connected than others. The hierarchical graph



(a)



(b)

Figure B.1 Example of (a) random graph and (b) transit-stub graph.

model attempts to capture this feature of real internetworks. In a N-level hierarchical graph, beginning with a connected graph (constructed by a random method), each node is iteratively replaced by a connected graph. The edges of the original graph is re-attached to nodes in the original graph. This is done N times and the resulting graph has a N-level hierarchy among nodes.

The transit-stub model is based on the hierarchical graph generation method that focuses on modelling the Internet topology. In this method, a connected graph is first generated using a random graph generation method. Each node of this graph represents a backbone topology of one transit domain. Then, within a transit domain, a number of random connected graphs are generated to represent different stub domains. These transit and stub domains represent transit and stub autonomous systems (AS) in the Internet. An example of a transit-stub graph is shown in Figure B.1b.

There are two transit-stub graphs used for modelling the physical network topologies in simulations in the thesis. Details of these graphs's parameters are listed in the Table B.1.

Each network edge is associated with two metrics: a routing policy cost and an Euclidean distance between the two nodes. The routing policy costs are automatically set by the GT-ITM package to represent the general routing characteristic of the Internet (Zegura et al., 1997):

- If node u and v are in the same domain, the shortest path between them remains within that domain.
- If node u in domain U and node v in domain V, the path from u to v goes from U, through zero or more transit domains, to V

The Euclidean distance is used for representing the link propagation delay. In a preprocessing process, the maximum end-to-end delay in the shortest path between any two nodes is determined. The maximum end-to-end delay is then adjusted to be equal to 300ms, which is based on the Internet delay analysis in (Bovy et al., 2002). In order to make this delay equal to 300ms, all link delays are multiplied by an arbitrary constant.

Parameter	Meaning	Value
T	Number of transit domains	3
N_t	Average nodes per transit domain	8
K	Average stub domains per transit node	3
N_s	Average nodes per stub domain	8
n	Total number of nodes	600
Node degree	$2m/n$, where m is the number of edges	4

(a) 600 nodes

Parameter	Meaning	Value
T	Number of transit domain	5
N_t	Average nodes per transit domain	10
K	Average stub domains per transit node	9
N_s	Average nodes per stub domain	11
n	Total number of nodes	5000
Node degree	$2m/n$, where m is the number of edges	3.9

(b) 5000 nodes

Table B.1 Parameters used for generating the transit-stub graph.

B.2 Physical Network and Virtual World

The model of the physical network and the virtual world has been described in Chapter 3. This section provides additional details to the description.

B.2.1 Physical Networks

The physical network consists of routers, ISP POPs and potential processing servers. ISP POPs and potential processing servers are randomly chosen from the set of nodes. All nodes can support routing function. The shortest path routing is implemented based on Dijkstra's algorithm which uses routing policy cost as the routing metric. After running these algorithms, the delay between nodes and the path connecting them are stored in several matrices.

Shortest path multicast is also implemented based on Dijkstra's algorithm. In this implementation, it is assumed that all nodes can support multicast routing. The purpose

of this multicast implementation is to store all multicast trees from each node to all other nodes. Hence, when multicast is used to deliver audio streams from each avatar to all other avatars in the background zone, the bandwidth cost saving of multicast can be determined and compared with the approach of using unicast.

Players are connected to ISP POPs. The number of players assigned to each ISP POPs is chosen randomly according to a uniform distribution. For example, in Chapter 4, the number of players connected to each ISP POPs is uniformly distributed in a range from 25 to 75, with an average value of 50.

B.2.2 Virtual World

There are three avatar grouping behaviours in the virtual world: loner, clan, and crowd. The graphic representations of these virtual worlds are provided in Chapter 3. This section describes additional details of parameters used for generating the virtual world.

The virtual world is modelled as a square size N. Each avatar has a coordinate (x, y), where $0 < x; y < N$. Avatar's coordinates are generated according to the following function.

- **Loners:** In each time that an avatar is placed, x and y are assigned with a random number in $(0, N)$, according to a uniform distribution.
- **Clans & Crowds:** Clans and Crowd are modelled as clusters. Firstly, a number of cluster centers are randomly generated in the virtual world in a uniform distribution (similar to the way loners are generated). In each cluster, avatars are placed around the center according to a normal distribution with the mean of 0. For example, let (a, b) be the coordinate of a cluster center, L be the cluster size, avatar's coordinates (x, y) are chosen as follows.

$$(x, y) = (a + normaldist(-L, L)), b + normaldist(-L, L)) \quad \text{(B.2)}$$

In particular, the above normal distribution has the mean of 0, the variance of 2, and the standard deviation of 1.4. Due to this normal distribution, in each cluster, the avatar density reduces when the distance to the cluster's center increases.

In all simulations, the interactive zone radius R_I and the background zone radius R_Z are chosen to be 5 and 20, respectively. The average avatar density depends on the size of the virtual world N and the size of cluster L. As an example, in a crowd virtual world that consists of 50 crowds, N and L are chosen to be 4000 and 200, respectively. In this case, the interactive avatar density (average number of avatars in the interactive zone) is 2.5. Varying L will result in different avatar densities.

B.3 Simulation Procedures

While some simulation parameters and procedures have been described in simulation setup in each simulation experiment, this section provides additional details of the simulation procedures.

In simulation experiments in each chapter, either the 600 node or 5000 node topology discussed earlier will be used. This network topology is fixed in the set of experiments in each chapter.

At the beginning of a simulation, a number of ISP POPs and potential processing servers are randomly chosen from the set of nodes. The result of this assignment as well as delays between nodes are stored in several matrices. Players are randomly assigned to ISP POPs. Hence, the virtual world is created based on an avatar aggregation method and the correlation parameter. Avatar position in the virtual world, clan or crowd, the relation between avatars and players, interactive zones, and background zones are stored in several matrices.

After that, optimization modules read the above parameters and calculate solutions for the server assignments. In the case of using Cplex, an input file is produced to capture all required parameters. This input file is read by Cplex and output file from Cplex are analyzed for representing a solution. In the case of using heuristics, the heuristics modules written in C++ read all these parameters and calculate solutions.

Finally, solutions are analysed and the results are outputted to the graph. Unless otherwise stated, each point of the result in a graph is associated with one simulation of a physical world and a virtual world with one server assignment scheme. When

several simulations are run for a set of result, error bars are plotted.

In each simulation, a set of result is often plotted based on varying some parameters. Key parameters that are varied in each simulation include: physical/virtual world correlation, avatar density, the number of servers or proxies, and the number of POPs. When a parameter is varied, each optimization module recalculates a new server assignment solution and the evaluation results are repeatedly obtained.