uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Dewan Tanvir Ahmed

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Architectural Challenges and Solutions for Peer-to-Peer Massively Multi player Online Games

TITRE DE LA THÈSE / TITLE OF THESIS

Shervin Shirmohammadi

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Abdulmotaleb El Saddik

Jörg Kienzle (McGill University)

Dorina Petriu

Thomas Tran

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Architectural Challenges and Solutions for Peer-to-Peer Massively Multiplayer Online Games

Dewan Tanvir Ahmed

Thesis Submitted to the Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the
Ph.D. degree in Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-61236-1
Our file Notre référence
ISBN: 978-0-494-61236-1

Abstract

Massively Multiuser Online Games (MMOG), now supporting millions of simultaneous participants on a regular basis, have become a significant contributor in human-to-human communications. While originally designed for games, they have now moved into serious realms of socialization, business, commerce, scientific experimentation, and others. As more and more people participate in these *massive* environments, the underlying infrastructure is starting to exhibit shortcomings that limit the progress, practicality, and applicability of MMOGs. This thesis explores various architectural challenges inherent in MMOGs and offers effective solutions in the context of a hybrid model. The key objective of this hybrid model, named ***M**assively **M**ultiuser **VI**rtual **S**imulation **A**rchitecture* (MM-VISA), is to form a stable and scalable collaboration platform that economically combines the resources of both servers and player peers, incorporating the advantages of a centralized architecture and a scalable Peer-to-Peer distributed system, which in turn leads to improved support for the participating masses.

Synchronous communication among massive number of users in an MMOG is a prime concern, and difficult and/or expensive to support. This massiveness causes challenges that cannot be solved with conventional techniques used in traditional collaborative environments. Massive number of players' frequent and random movements in the virtual environment and zone-switching can easily break synchronous communication and cause substantial strain on the underlying system, networking, and server

infrastructure. To alleviate such problems, this thesis proposes a model consisting of interest-driven zone crossing, dynamic shared regions, clustering of players based on their attributes, multilevel multiphase load-balancing with several plug-able solutions, hybrid routing based on a combination of centralized and Peer-to-Peer (P2P) networking, and interest-management techniques considering dynamics of the area of interest and graphical computing. It is then revealed that the model significantly improves overall system performance and enhances infrastructure stability in terms of load, network overlay, and other performance characteristics.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Prof. Shervin Shirmohammadi, for his continuous unwavering support of my doctoral study and research, for his patience, motivation, enthusiasm, and immense knowledge. It has been an honor to be his first Ph.D. student. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. The joy and enthusiasm he has for my research was contagious and motivational for me, even during tough times in the Ph.D. pursuit. His guidance helped me all through the researching and writing of this thesis. I cannot imagine having a better advisor and mentor for my Ph.D. study.

Besides my advisor, I would like to thank the rest of my thesis committee for their insightful comments and valuable time. I would also like to acknowledge my colleague Ihab Kazem for the collaboration and discussion, and implementing a portion of the prototype. I would also like to thank Jimmy Bonny for his OPNET simulation.

My parents, Dewan Munir Ahmed and Rehana Khatun, have been a continuous source of love, support and encouragement all throughout this PhD process. Special thanks go to my brothers, Dewan Hasan Ahmed and Dewan Imran Ahmed, and relatives for cheering me up at all times. Last but not least, I am grateful to my dearest wife, Farah Jasmeen, for her belief and encouragement to follow my dreams ...

Dedication

I dedicate this thesis to my parents....

List of Abbreviations

ALM	Application Layer Multicasting
AOI	Area of Interest
AoIM	Area of Interest Management
BFS	Breadth-First Search
CDF	Cumulative Distribution Function
CIM	Collaborative Interaction Management
DoD	Departments of Defense
DS-ALM	Dominating Set based Application Layer Multicast
DVMRP	Distance Vector Multicast Routing Protocol
ESM	End System Multicast
ETE	End-to-End
FPS	First Person Shooter
HDSP	Hybrid Distributed Simulation Protocol
IGMP	Internet Group Membership Protocol
ISP	Internet Service Providers
LCVE	Large-scale Collaborative Virtual Environment
LSVE	Large-Scale Virtual Environments
MMLB	Multilevel Multiphase Load Balancing
MMOG	Massively Multiuser Online Games
MMORPG	Massively Multiplayer Online Roleplaying Games

MM-VISA	Massively Multiuser VIrtual Simulation Architecture
MST	Minimum Spanning Tree
NPC	Non-playing Components
NVE	Networked Virtual Environment
P2P	Peer-to-peer
QoS	Quality of Service
RTT	Round-trip Time
SCORE	Scalable Multicast-based Communication Protocol
SIM	Scene Interaction Manager
SL	Second Life
SLA	Service Level Agreement
SPT	Shortest Path Tree
TIM	Task-Oriented Interaction Management
VR	Virtual Reality

Contents

Abstract ii

Acknowledgements iv

Dedication v

List of Abbreviations vi

List of Figures xiii

List of Tables xviii

1 Introduction 1

1.1 Virtual Environments and Online Games 1

1.2 Motivation 5

1.2.1 Applications - Simulation to Entertainment 5

1.2.2 Scalability 5

1.2.3 Communication Architecture 6

1.2.4 Area of Interest Management 8

1.3 Research Problem and Direction 10

1.4 Research Contributions 12

1.5 Scholastic Output and Achievements 12

1.6 Road Map 18

2 Background and Related Work **19**

2.1 Zoning and Area of Interest 20

2.1.1 Multiple Zones and Their Shapes 20

2.1.2 Area of Interest Management 21

2.1.3 Discrete View versus Continuous View 24

2.1.4 Geographic versus Behavioral Modeling 24

2.2 Networking and Scalability for MMOGs 25

2.3 Application-Layer Multicasting 30

2.4 Load Balancing 33

3 Application-Layer Multicasting and P2P Communications **35**

3.1 Trade-off: ALM vs. IP Multicast 35

3.2 ALM Protocol Design 40

3.2.1 Application Domain 41

3.2.2 Deployment Level 42

3.2.3 Group Management 44

3.2.4 Routing Mechanisms 49

3.2.5 Degree-Constraint Routing 52

3.3 Challenges 54

4 The Hybrid MMOG Architecture **57**

4.1 Architectural Overview and Assumptions 57

4.2 Clustering of Players 61

4.3 Intra-Zone Communication 63

4.3.1 General Policy and Node Registering 64

4.3.2 Mesh Construction 66

4.3.3 Data Delivery Path Based on Dominating Set 67

4.3.4 Handling the Ad hoc Nature of Nodes 72

4.4 Message Overhead Reduction 75
4.5 Interest-driven Zone-crossing 77
4.6 Visibility Issues 80
4.7 Seamless Player Hand-Off 82
4.8 Summary 84

5 Hotspots and Load-Balancing Mechanisms 85
5.1 Load Balancing using Two Level Partitioning 86
5.1.1 On-the-Fly Two-Layer Partitioning 86
5.1.2 Rejecting a Load 88
5.1.3 Multiple Hotspots 88
5.1.4 Undo Mode 90
5.1.5 Discussion 90
5.2 Load Balancing: Multilevel Multiphase approach 91
5.2.1 New Load Definition 92
5.2.2 Identifying the Loaded Server 93
5.2.3 Phase 1: Load Balancing for Top-level Microcells 94
5.2.4 Phase 2: Exploiting Buffer Region to Release Load 95
5.2.5 Phase 3: Load Handling through Deep-level Partitioning . . 96
5.3 Load Balancing for Non-Uniform Zones 99
5.3.1 The Algorithm 99
5.3.2 Zone Merging 102
5.3.3 Zone Capturing 103
5.3.4 Dynamic Cut Movement 103
5.4 Summary 103

6 Quality and Design Issues 105
6.1 Dynamic Area of Interest Management 106

6.1.1 The Proposed Approach 106
6.1.2 Interest Management Classification 110
6.1.3 Comparison 114
6.2 Expedited State Sharing 116
6.2.1 Homogeneous Case 119
6.2.2 Heterogeneous Case 120
6.3 Quality Control 120
6.3.1 Problem Formulation 121
6.3.2 Peer Potentials 122
6.3.3 Physical Position Approximation 122
6.3.4 Virtual Position and its Impact 123
6.3.5 Message Redirection Procedure 123
6.3.6 Quality Control 124

7 Evaluation **127**
7.1 Effectiveness of Intra-Zone Communication 128
7.1.1 Technical Considerations 130
7.1.2 Real-world Game Traffic 130
7.1.3 Simulations: DS-ALM vs. NICE 135
7.2 Stability Improvement 145
7.3 Measurements of Interest-driven Zone-crossing 148
7.4 Load-Balancing Performance 152
7.5 Evaluating Dynamic AoI Management 158
7.5.1 Advantages of the Architecture 159
7.5.2 Minimizing Communication Overhead and Improving Scalability 159
7.5.3 Dynamic AoI for Various Movement Patterns 160
7.6 Expedited State-Forwarding Improvements 162

7.7 MMOG Quality Improvement 166

8 Validation **167**

8.1 End-to-end Delay Measurement 167

8.2 Overlay Stabilization Time Measurement 168

8.3 Impact of Buffer Zone Size 169

8.4 Number of Tree-level 170

8.5 Validating Expedited State Forwarding Concept 171

9 Conclusion and Future Directions **174**

9.1 Contributions 175

9.2 Future Work 176

Bibliography **191**

List of Figures

1.1 Players interacting in Ultima Online, a classic MMORPG 2

1.2 The concept of (a) IP multicast and (b) ALM 9

2.1 Two versus multiple zones layout, with a player moving across zones 20

2.2 Hexagonal versus circular zone shapes 21

2.3 The aura-nimbus model 22

2.4 Region-based area of interest 23

3.1 (a) Application layer multicast (b) IP multicasting scenario 36

3.2 (a) Sample overlay topology (b) an overlay multicast tree 37

3.3 (a) A physical topology (b) IP multicast tree constructed by DVMRP (c) ALM concept (d) End-system overlay network 38

3.4 (a) Proxy-based deployment of ALM (b) End-system ALM 43

3.5 (a) A mesh: a network topology with many redundant interconnections between network nodes (b) Initial tree (c) Lopsided Tree . . . 46

3.6 (a) A graph with link costs (b) Shortest path tree (b) Minimum spanning tree 50

3.7 A hierarchical cluster of nodes with cluster size 4 52

4.1 Number of zones covered by a visibility circle in hexagonal, square, and triangular zones (respectively from left to right). 58

4.2 The top level hybrid architecture 59
4.3 The clustered ALM, message propagation and isolation scenery . . . 63
4.4 Controlling the diameter of the tree from master's perspective 65
4.5 Random choice leads to disconnected graph 67
4.6 Connected dominating set formation: (a) Initially all nodes are non-gateway nodes, (b) A node becomes gateway if it has two unconnected neighbors, (c) Rule-1, (d) Rule-2 71
4.7 The ignore set buildup 72
4.8 Handling node departures 74
4.9 Key and regular messages in the timeline 75
4.10 (a) Propagation of key message (b) Prune activation message (c) Filtering 76
4.11 Impact of zone crossing 78
4.12 (a) Hexagonal regions with check-in and check-out radii with dynamic adjustment of zone marks (b) Controlling of frequent zone crossings 79
4.13 Solution to visibility problem: (a) a simplest approach (b) a smarter approach 81
4.14 State sharing leads to smooth zone crossing 83

5.1 Two-layer partitioning 87
5.2 Multiple-hotspot scenario 89
5.3 The layout of microcells and a random microcell distribution to servers 93
5.4 Stepwise deep-level partitioning for load balancing 97
5.5 (a) Virtual world partitioning (b) Zone naming 100
5.6 The cut movement 102

6.1 The symmetric and asymmetric relation of interest 107
6.2 The safety margin of a convex hull 110

6.3 Euclidean distance algorithm for interest management 111
6.4 Square tile algorithm for interest management 111
6.5 Hexagonal tile algorithm for interest management 112
6.6 Ray visibility algorithm for interest management 113
6.7 State information distribution through binary tree 117
6.8 State distribution through expedited mechanism 118
6.9 Core model of a player 119
6.10 The quality control algorithm in MMOGs 125

7.1 MMORPG client traffic from ShenZhou: exponential distribution, mean packet inter-arrival time = 550ms 132
7.2 FPS client traffic from Quake: extreme distribution, mean packet inter-arrival time = 40ms 132
7.3 Inbound traffic for a hybrid MMORPG client with 10 neighbors (exponential distribution, mean packet inter-arrival time is 550ms for each neighbor) 133
7.4 Inbound traffic for a hybrid FPS client with 10 neighbors (extreme distribution, mean packet inter-arrival time is 40ms for each neighbor) 133
7.5 The cumulative distribution function (CDF) of the packet size of (a) a P2P player (b) one player of a client-server model 134
7.6 The worst-case load comparison between NICE and MM-VISA using MMORPG traffic 135
7.7 The worst-case load comparison between NICE and MM-VISA using FPS traffic 136
7.8 Maximum diameter in terms of delay 137
7.9 Maximum diameter in terms of hops 137
7.10 Average ETE distance in terms of time 138
7.11 Average ETE distance in terms of hops 139

7.12 The comparison of stretch 139
7.13 Comparison of diameter in terms of time (NICE & DS-ALM) 140
7.14 Effect of degree on the number of hops 141
7.15 Comparison of diameter in terms of hop with the different degrees . 141
7.16 Average ETE evolution (NICE & DS-ALM) 142
7.17 Average number of hops evolution with the degree (NICE & DS-ALM) 143
7.18 Evolution of stretch with the degree (NICE & DS-ALM) 144
7.19 Simulation layout 144
7.20 The avatar movement model 145
7.21 The analysis of entity typing concept 147
7.22 Improvement ratio with respect to no clustering mechanism 147
7.23 Various zone crossing models & movement patterns 148
7.24 A random square-step walk 149
7.25 Another random walk 150
7.26 Comparison of zone crossing approaches for a player 151
7.27 Message overhead reduction with respect to degrees 151
7.28 The load based on player counts 153
7.29 The load based on packet counts 153
7.30 The density of players across zones 154
7.31 The standard deviation of the processed packets by the servers' overtime with no load-balancing mechanism 155
7.32 The standard deviation of processed packets by the servers' overtime using the first phase of MMLB 155
7.33 The evaluation of MMLB against the generic load balancing technique 156
7.34 The evaluation of MMLB for different load handover magnitude . . 157
7.35 The comparison of MMLB and adaptive zone shaping with respect to hotspots 158
7.36 Two players need to be contacted to adjust AoI 159

7.37 Distribution of AoI tracking responsibility in a group of 40 players . 160
7.38 Performance of dynamic AoI for different group sizes - 10,000 movement steps 161
7.39 Comparing theories by measurement 162
7.40 The measurement of average time to share state information 163
7.41 The measurement of maximum time required to share state information 164
7.42 The drop-off of the maximum latency 164
7.43 The maximum latency improvement for various group sizes 165
7.44 The average latency with 95% confidence interval for difference group sizes 165

8.1 A scenario to measure overlay stabilization time 168
8.2 The common area between two zones with three random paths . . . 170
8.3 Performance improvement due to the use of common area 171
8.4 The path used for quality measurement due to expedited state forwarding 172

List of Tables

3.1 Conceptual comparison of IP Multicast and ALM 39

4.1 Routing table for Gateway Node x 77
4.2 Message exchange reduction for different buffer sizes 83

5.1 Load release at different stages for various buffer sizes against current load 96
5.2 Load shedding comparison 98
5.3 The cut movement rules 102

6.1 Comparison of different interest management algorithms 115

7.1 Protocol similarity 129
7.2 The simulation parameters used for stability testing 146
7.3 A comparison of crossing in different approaches 148

8.1 Reflecting the end-to-end delay on the ALM overlay network 168
8.2 The overlay stabilization time 169
8.3 The measured end-to-end delay between regions 173

Chapter 1

Introduction

1.1 Virtual Environments and Online Games

Games are a form of entertainment, a source of excitement, fun and socialization. *Massively multiplayer online (role-playing) games*, MMOGs or MMORPGs, are a new genre of online games that emerged with the introduction of Ultima in 1997 (Figure 1.1). An MMOG is a kind of online computer game built on the participation of hundreds of thousands of players in a virtual world. An MMOG such as *World of Warcraft* or *Quest* can have millions of subscribers. According to an announcement on the game's official website, a record 45,186 gamers have gathered together in *EVE Online*'s virtual environment, breaking the previously announced record of 35,000 gamers [1] .

A fascinating imaginary environment is the key to a successful MMOG that will attract players. These players participate in activities and races, enhance their resources, exercise their power and improve their skills by carrying out various missions. Each player has a graphical representation of the virtual world and controls an avatar, which can perform different actions. Examples of such actions include moving the

[1] EVE online (2009): http://www.eve-online.com/

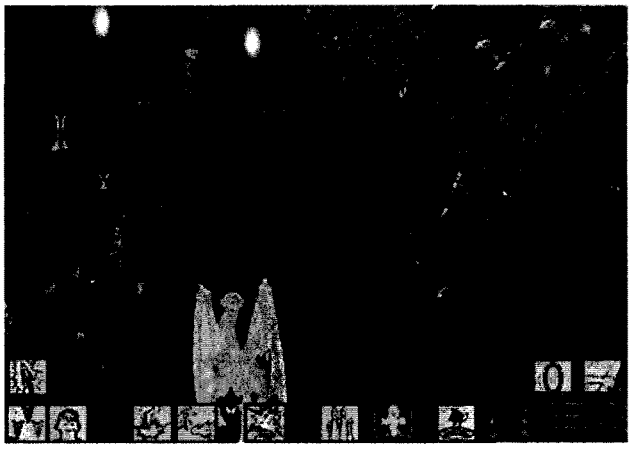

Figure 1.1: Players interacting in Ultima Online, a classic MMORPG

avatar (path-finding), picking up objects, or communicating with other players. The challenge of an MMOG is to administer a massive number of connected players and present them a consistent view of the game world.

In addition to entertainment, these games offer "*the reward of being socialized into a community of gamers and acquiring a reputation within it*" [DYNM06]. In many cases, MMOGs have become addictive for players. Ducheneaut et al. present an inclusive case study on how World of Warcraft is designed to catch the interest of subscribers, who also become audiences for other players' in-game performances, providing an easy and accessible source of information and chitchat [DYNM06]. In this sense, playing the game is like being "alone together"-the player is surrounded by others for "spectator experience" and an impression of social presence. Klastrup goes further, adding a psychological factor to these games [Kla06]. He studied how players' culture and stories enforce a "heroic" approach to a problem, e.g. death, directly and indirectly, serving as a guide to new players, teaching them how to "do better" both as social players and gamers.

Online games have achieved popularity due to increasing broadband adoption

among consumers. Relatively cheap high-bandwidth Internet connections allow large numbers of players to play together. With the advancement of computer graphics and artificial intelligence, the success of online gaming is growing dramatically. It has now become a profitable sector to vendors. New players are always joining online games to share the gaming experience with fellow players in alluring virtual worlds such as Sony's *EverQuest*, Valve's *Half-Life*, and Blizzard's *World of Warcraft.* According to a general report from game companies, the number of online players at any given time is between 6,000 and 10,000, bringing in a powerful US$1 billion in subscription revenue in 2004 [DFC03].

Interestingly, the most valuable Internet company in China is an MMOG, clicking in at US$2.0 billion in market cap. Once considered a small business, multiplayer gaming is growing sharply. It has an enviable business prospect for many reasons like recurring revenues, competitiveness, social networking effect, real competition, and the amount of time engaged. For most online games, the experience of gaming is compelling for incremental users. It appears *Metcalfe's Law of self-reinforcing* form is alive in many MMOGs. The big criticism of MMOG is that it is a *hit* business like Hollywood. A closer look will reveal that the average life of a successful MMOG is around five years.

The growth and evolution of MMOGs is confirmed by technical data and measurements from different organizations and surveys. For the past few years, studies have revealed that online games are becoming a major contributor to Internet traffic. DFC Intelligence forecasts that the worldwide online game market will expand from $4.5 billion to over $13 billion in the period from 2006 to 2011. More interestingly, in that period, the total number of online gamers in the 20 leading online gaming countries is expected to increase by 51%. A meticulous analysis by In-Stat/MDR reports that approximately 9% of the traffic sent back and forth over the U.S. backbone was due to online gaming in 2002, and this is estimated to reach nearly 30% by 2008 [Man04]. More recently, the NPD Group reports that 59% of the total U.S.

population (ages 2 years old and higher) play games, with 56% of them doing so online, leading to almost a third of the population playing online games [2] . Although similar studies are not available for Canadian gaming traffic, it is estimated that the percentages in Canada are about the same if not higher due to the very high rate of home Internet penetration in Canada, and the fact is that Canada is consistently ranked in the top five countries in per-capita spending on computer games.

MMOGs are now widely used for socializing, business, commerce, scientific experimentation, and many other practical purposes. One could say that MMOGs are the "killer app" that brings MMVE into the realm of eSociety. This is evident from the fact that real companies are opening "virtual branches" in these online games, such as jean manufacturers, IBM, and CNN, to name a few. In fact, IBM has ready launched a free multiplayer online game, *PowerUp*, which challenges teens to save a fictional planet from ecological disaster. Virtual currencies such as the Linden [3] (or L$) in *Second Life* are already being exchanged for real-world money . Similarly, virtual goods and virtual real estate are being bought and sold with real-world money. Massive numbers of users spend their time with their fellow players in online games like *EverQuest*, *Half-Life*, *World of Warcraft*, and *Second Life*. *World of Warcraft*, for example, has over 10 million users with a peak of over 500,000 players online at a given time. There is no doubt that MMOGs and MMVEs have the potential to be the basis of any eSociety in the near future, because they bring the massiveness, awareness, and interpersonal interaction of real society into the digital realm.

The biggest surprise with online games is their diversity. Compared with the traditional video and PC game industry, online games offer many more types of games, more business model options and greater international appeal. And finally, online games are one of the great "sticky" content categories. Many consumers subscribe to one service over another. The business is always active.

[2] NDP Group, "Online Gaming 2008", http://www.npd.com

[3] 1 US$ = 266.50 L$, with roughly US$ 1.5 million exchanged daily

1.2 Motivation

1.2.1 Applications - Simulation to Entertainment

MMOGs are similar to the generic massively multiuser simulations that have existed for decades, most notably combat training simulations used by Departments of Defense (DoD) around the world, and more recently, disaster management applications, emergency planning simulators, etc. These have reached their current state because of their significant impact on virtual training in high-risk situations as well as their ability to interpret the real and simulated results in extraordinary circumstances such as natural disasters or terrorist attacks.

Military simulations are seen as a useful way to develop tactical solutions. The scope of simulations has widened to include not only military but also political and social factors. Defense ministries and aviation companies are primary users of this technology, which offers a virtual world in which parties can interact and collaborate in real time. A scalable architecture that is fully deployable over the Internet is in demand. MMOGs, a system for entertainment and fantasy, are a variant of such environments.

The motivation behind this research is therefore clear: MMOGs are here to stay, and will be used not only for gaming but also for many other purposes. Therefore, an architecture that could efficiently support such environments would be of a scientific value and a contribution to the field.

1.2.2 Scalability

The development of an MMOG faces many challenges. Multiuser simulations and MMOGs introduce hard challenges to the system designers. The most important challenge is scalability, which is a desirable property of a system that indicates its ability to either handle growing amounts of work in a graceful manner, or to be readily

enlarged. This is a critical issue to consider when designing large-scale simulators and MMOGs, as it is a complex function of the other components of the system that requires regular exchange of update messages among the participants.

Online games also have a set of other requirements - consistency, responsiveness, reliability, security, and persistency [SKH01]. Real-time applications like networked games require observing the effect of an action in time. Network latency and processing delays, however, make this difficult to achieve. It is true that latency tolerance usually varies from game to game and is typically limited to a value between 100ms and 1000ms [CC06]. This value depends on game perspectives (i.e. first-person or third-person), game genres (i.e. racing or role playing game), and the sensitivity of actions. The system's scalability depends on servers' and clients' available bandwidth, types and frequencies of activities, and as well as players' density in a given region. The key factor for scalability is whether resource usage is bounded both at clients and servers. Reliability may be characterized by fault-tolerant capabilities in case of accidental/intentional software and hardware failures. In such cases, normal operations must be resumed as early as possible without a noticeable disturbance.

1.2.3 Communication Architecture

Commercial MMOGs use the client-server architecture with a single authentic server designed to support the game logic. In addition to the bottleneck problem, the client-server architecture is expensive to deploy and maintain. For example, SL (*Second Life*) has approximately 5000 servers to support its virtual space. Such expensive deployment issues, as well as complex maintenance demands, are common in bettering gaming experience, performance and administrative control. In MMOGs, the server pool regulates game traffic using the zoning concept and makes its implementation more convenient. Practically, the communication structure within a zone is similar to the Internet multicast structure, not client-server, because of the players' common

interest in the game logic. IP multicast, which was originally proposed for group communication, can be an ideal solution for this purpose. But it is a well-known fact that IP multicast is not fully deployable on the wide-scale Internet, even in the future with IPv6 [ESRM03][HASG07][DLL+00]. Hoplon's Taikodom, which has managed to support 700 users in one zone, with 50 of them engaged in battle [Gui08]. However, Taikodom uses an IBM z10 supercomputer to achieve that! This is not an affordable solution for all MMOGs.

Network lag is another well-known problem that affects the performance of the system. In online games, when a player interacts with other players, the updated information must be sent to all participants. Because of networking limitations and traffic conditions, some of these updates might be delayed or lost. Much research has been conducted to overcome the networking limitations and provide a better distributed system. Some of these studies provide receiver-initiated [PK99] and selectively reliable transport protocols [Pul99] that can be used to deliver important messages with a high degree of reliability, while others use sender-initiated approaches, transmitting key updates with guaranteed reliability [SG01]. The IEEE DIS standard [IEE98] has also been successfully used in a controlled environment with vast resources, mostly for military simulations. These approaches are based on IP multicasting, and, although they achieve good results in an Intranet environment, they are not readily deployable on the Internet.

A peer-to-peer (P2P) computer network is a network which generally relies on computing power and bandwidth of the users rather than one or more servers. Peer-to-peer architecture has self-scalable properties, but considering its business issues and quality concerns, it is apparent that pure peer-to-peer architecture is not a viable solution. At present, different mixed architectures are being proposed using peer resources, but the practical deployment hurdles are not yet fully overcome. Current designs (research-oriented), however, try to incorporate client- and server-side resources in a peer-to-peer fashion to address different challenges such as scalability,

responsiveness, and persistence [KLXH04][IHK04][YV05]. The peer-to-peer community strongly believes that development of online games over such a platform would be valuable in terms of deployment cost, and of performance, in some sense, through reduced latencies.

1.2.4 Area of Interest Management

The game space of MMOGs contains plenty of information, but a single player needs to be informed of only a small subset of that information. For online games, *Area of Interest Management* (AoIM) is a technique used to reduce communication overhead. AoIM methods intelligently determine useful information for a player and block other irrelevant information. For example, the area of interest of an avatar in an MMOG is the set of avatars and *non-playing components* (NPC) with whom it interacts within its neighborhood. Since a game's virtual world is large, filtering out irrelevant information is a fundament requirement for a scalable system. Thus, relaying relevant information to each player is an effective way to approach messaging in online games. But the need for an efficient support of one-to-many and many-to-many applications led to the proposed implementation of multicasting on the global inter-network called IP Multicast [DC90].

An IP multicast-capable network can allow one or more sources to efficiently send data to a group of receivers, wherein the source transmits only one copy of a packet and the appropriate network nodes, i.e. routers, efficiently make duplicate copies for each receiver as needed (Figure 1.2a). After a decade of research into the various issues of IP multicasting, such as routing, group management, address allocation, authorization and security, quality of service (QoS) and scalability, the widespread deployment of IP multicast on the global inter-network has struggled with technical, administrative and business-related issues [DC90]. Internet Service Providers (ISPs) rarely allow home users to be a part of an IP multicast session. Therefore, there have

been recent proposals for alternative group communication services that would grow out of the IP multicast model. El-Sayed et al. survey such proposals [ESRM03] and present multicasting approaches alternative to classic IP multicasting. These include the use of reflectors, permanent tunneling (e.g. MBONE), relying on specific routing services such as IPv6, and application-layer multicasting or automatic tunneling.

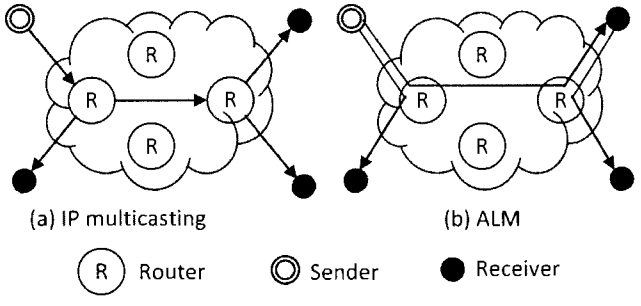

Figure 1.2: The concept of (a) IP multicast and (b) ALM

Arguably, the most influential development to encourage and bring about the idea of Application Layer Multicasting (ALM) has been the tremendous success of peer-to-peer file-sharing applications, such as *Napster*, and its successors, such as *Kazaa*. Millions of home Internet users with limited dial-up bandwidth and DSL connections could cooperate together in order to distribute gigabits of media content on the global network. Its success therefore encouraged other forms of media distribution services such as teleconferencing and media on demand using end systems, acting as cooperating peers. In ALM, data packets are replicated at end-hosts rather than at routers, as shown in Figure 1.2b. As the ALM does not require any special infrastructure support, it is fully deployable on the Internet. However, an ALM does come with trade-offs: more bandwidth and delay (compared to IP multicasting). But it has been shown that ALM-based algorithms can have "acceptable" performance penalties with respect to IP multicasting and other practical solutions [CRZ02].

1.3 Research Problem and Direction

In this thesis, a hybrid MMOG architecture, Massively Multiuser VIrtual Simulation Architecture (MM-VISA) is presented that provides the properties of a centralized architecture, while exploiting P2P communication to achieve scalability. This "hybrid" (centralized combined with P2P) MMOG architecture divides the virtual world into several manageable zones where each zone covers the players in a given vicinity. This division of the game into zones is a common practice and not exclusive to our architecture. The rationale behind zoning is that players are only interested in what goes around in their own vicinity. In our architecture, each zone has a zone master responsible to build a P2P overlay network within its zone for intra-zonal communication among players. This P2P communication model can reduce traffic load on the servers, and decrease deployment and maintenance cost. Together, the set of zone masters forms a top-level management mechanism that regulates the operation of the MMOG.

As the game world is broken into smaller manageable zones, a player can move from its current zone to a neighboring zone (called zone crossing) causing the restructuring of the overlay network. This restructuring will change the P2P overlay and cause problems for routing messages. To limit such problem, the proposed architecture uses players' gaming characteristics, such as velocity, to group them into clusters, where players in each cluster have similar characteristics. With clusters in each zone, a leaving player only affects the players in that cluster. Thus, this structure reduces the zone-crossing problem from the whole zone to only a single cluster. New methods like interest-driven zone crossing and dynamic shared regions between adjacent zones, which will be explained in Chapter 4, are proposed that help making informed decisions to solve the above problems more efficiently.

Due to massive number of players, MMOG applications require much network bandwidth to function properly. In a distributed MMOG server architecture, the

server nodes may become overloaded by the high number of players. The main resource to consider here is the number of packets produced by the players; the more players join the same server, the more packets are produced for that server, demanding more bandwidth; this is the current bottleneck for MMOGs. To address this problem, this thesis presents load-balancing algorithms for both uniform and non-uniform zonal MMOGs. In uniform zones, zones' shape and size are the same; like hexagons of the same size. On the other hand, in non-uniform zones, the size can be different like rectangles of different sizes. The proposed load-balancing schemes identify a loaded server in terms of either the number of players or packets processed per unit time, and then move the load to other servers considering communication overhead and P2P overlay restructuring. The proposed Multilevel Multiphase Load Balancing (MMLB) method introduced in chapter 5 is designed for uniform zones. MMLB reduces load in a step-by-step manner, and avoids problems associated with current load balancing schemes. We also present a novel load balancing scheme for non-uniform zones using a bisection procedure that does not adhere to any predefined zone size; i.e., zone sizes are flexible and can be determined dynamically.

Broadcasting all update messages to every player is not a viable solution to maintain a consistent game world. To successfully overcome this challenge, multiplayer online games need to employ sophisticated interest management techniques as explained earlier. Here we propose an area of interest management technique for a peer-to-peer architecture. This technique assigns interest management duties to a subset of players for each AoI. As players move in the virtual world, the area of interest also changes. So, the subset of players performing the interest management duties needs to be redefined. Apart from this, as an improvement, several plug-able solutions are proposed such as expedited state dissemination, opportunistic state forwarding to comply with hard time constraints, and others which will be explained in Chapter 6.

1.4 Research Contributions

In short, the main contributions of the thesis are a comprehensive MMOG architecture consisting of the following novelties:

- The thesis presents a hybrid communication middleware for MMOGs that exploits P2P technology to allow efficient player communication within a zone, and between zones. To improve performance and achieve P2P network stability, it proposes player clustering and interest-driven zone crossing.
- Three load-balancing techniques are presented - two for uniform and one for non-uniform zonal MMOGs.
- This thesis proposes an area of interest management technique for MMOGs in the context of a P2P architecture.
- In an MMOG, quick dissemination of the current state of players and objects is of outmost importance. This thesis also proposes an expedited state sharing mechanism among players.
- The thesis also proposes an algorithm that transports messages between players based on their virtual and geographical (real-world) positions, improving the gaming experience among players.

1.5 Scholastic Output and Achievements

In addition to meeting its objectives as described above, this research undertaking has also lead to a variety of scholastic achievements and publications, as listed below.

Awards

1. OCRI Student Researcher of the Year, 2009
2. Inscribed on the 2007-2008 Faculty of Engineering's Dean's Honour List
3. Best Poster Award in Computer Science, Faculty of Engineering Research and Graduate Studies Day, 2008
4. Student Travel Grant Award, IEEE International Instrumentation and Measurement Technology Conference, Victoria, BC, Canada, 2008
5. Best Paper Award in IEEE WETICE COPS workshop, Paris, France, 2007
6. Research travel grants, University of Ottawa (3 times)

Refereed Journals (published)

1. Dewan Tanvir Ahmed, Shervin Shirmohammadi, Jauvane C. Oliveira, "A Hybrid P2P Communications Architecture for Zonal MMOGs", Multimedia Tools and Application (Springer Netherlands), vol. 45, I(3), pp. 313-345, 2009
2. Shervin Shirmohammadi, Ihab Kazem, Dewan Tanvir Ahmed, Madeh El-Badaoui, Jauvane C. Oliveira, "A Visibility-Driven Approach for Zone Management in Simulations", SCS Simulations, V. 84(5), 215-229, 2008
3. Mojtaba Hosseini, Dewan Tanvir Ahmed, Shervin Shirmohammadi, Nicolas D. Georganas, "A Survey of Application-Layer Multicast Protocols", IEEE Communications Surveys and Tutorials, vol. 9, I(3), pp. 58-74, 2007

Refereed Journals (under review)

1. Dewan Tanvir Ahmed, Shervin Shirmohammadi. "A Load Management System for Massively Multiplayer Online Games", Emerging Systems with Advanced Information Networking Technologies, a special issue of IEEE Systems journal, submitted - under review, 2009

Book Chapters

1. Dewan Tanvir Ahmed, S. Shirmohammadi. "Zoning Issues and Area of Interest Management in MMOGs", Handbook of Digital Media in Entertainment and Arts, Borko Furht, Springer, pp. 175-196., and ISBN: 978-0-387-89023-4, 2009

2. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Networking for Massively Multiuser Online Gaming", Encyclopedia of Multimedia, pp. 664-670, Borko Furht, Springer, ISBN: 978-0-387-74724-8, 2008

3. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Mobile P2P Computing", Encyclopedia of Wireless and Mobile Communications, pp. 751-758, Borko Furht, Taylor & Francis, ISBN: 1420043269, 2008

4. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Multicasting in Mobile Ad Hoc Networks", Encyclopedia of Wireless and Mobile Communications, pp. 546-555, Borko Furht, Taylor & Francis, ISBN: 1420043269, 2008

Conference and Workshop Papers

1. Dewan Tanvir Ahmed, Shervin Shirmohammadi, " An Algorithm for Measurement and Detection of Path Cheating in Virtual Environments", Proc. IEEE Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Istanbul, Hong Kong, May 2009

2. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Model and Measurement of MMOG Time-Constraint Relaxation Algorithm", Proc. IEEE International Instrumentation and Measurement Technology Conference, Singapore, May 5-7, 2009

3. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Intelligent Path Finding for Avatars in Massively Multiplayer Online Games", Proc. IEEE Workshop on

Computational Intelligence in Virtual Environments, in Proc. IEEE Symposium Series on Computational Intelligence, Nashville, TN, USA, March 30 - April 2, 2009

4. Dewan Tanvir Ahmed, Shervin Shirmohammadi, " A Dynamic Area of Interest Management and Collaboration Model for P2P MMOGs", Proc. IEEE Int. Symposium on Distributed Simulation and Real Time Applications, Vancouver, BC, Canada, October 2008

5. Dewan Tanvir Ahmed, Shervin Shirmohammadi, " A Microcell Oriented Load Balancing Model for Collaborative Virtual Environments", Proc. IEEE Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Istanbul, Turkey, July 2008

6. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Model and Measurement of State Dissemination in MMOGs", Proc. IEEE International Instrumentation and Measurement Technology Conference, Victoria, BC, Canada, May 2008

7. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "An Expedite State Dissemination Mechanism for MMOGs", Proc. International Symposium on Parallel Architectures, Algorithms, and Networks, Sydney, Australia, May 2008

8. Razib Iqbal, Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Distributed Video Adaptation and Streaming for Heterogeneous Devices", Proc. IEEE Workshop on Mobile Peer-to-Peer Computing, Hong Kong, China, March 2008

9. Dewan Tanvir Ahmed, Shervin Shirmohammadi, Jauvane C. Oliveira, "Performance Enhancement in MMOGs Using Entity Types", Proc. IEEE Int. Symposium on Distributed Simulation and Real Time Applications, Chania, Crete Island, October 2007

10. Ihab Kazem, Dewan Tanvir Ahmed, Shervin Shirmohammadi, "A Visibility-Driven Approach to Managing Interest in Collaborative Virtual Environments with Dynamic Load Balancing", Proc. IEEE Int. Symposium on Distributed Simulation and Real Time Applications, Chania, Crete Island, October 2007

11. Dewan Tanvir Ahmed, Shervin Shirmohammadi, Jauvane C. Oliveira, "Improving Gaming Experience in Zonal MMOGs", Proc. ACM Multimedia, Augsburg, Germany, September 2007

12. Dewan Tanvir Ahmed, Shervin Shirmohammadi, Jauvane C. Oliveira, "State Management in Large Scale Group Communication", Proc. IEEE Int. Conference on Signal Processing and Communication, Dubai, UAE, November 2007

13. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "A Framework for Provisioning Overlay Network Based Multimedia Distribution Services", Proc. IEEE International Conference on Multimedia and Expo, Beijing, China, July 2007

14. Dewan Tanvir Ahmed, Shervin Shirmohammadi, A. El Saddik, "A Dominating Set Based Peer-to-Peer Protocol for Real-Time Multi-Source Collaboration", Proc. IEEE Workshop on Collaborative P2P Information Systems, Paris, France, June 2007

15. Dewan Tanvir Ahmed, Shervin Shirmohammadi, Jauvane C. Oliveira, "Supporting Large-Scale Networked Virtual Environments", Proc. IEEE Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Ostuni, Italy, June 2007

16. Choudhury A. Al Sayeed, Dewan Tanvir Ahmed, Akbar G. P. Rahbar, "Hybrid Maximal Matching for Input Buffered Crossbar Switches," Proc. IEEE/ACM Conference on Communication Networks and Services Research, Fredericton, NB, Canada, May 2007

17. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Architectural Analysis of Multicast Routing Protocols for Wireless Ad Hoc Networks", Proc. IEEE International Conference on Networking, Martinique, April 2007

18. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Design Issues of Peer-to-Peer Systems for Wireless Ad Hoc Networks", Proc. IEEE International Conference on Networking, Martinique, April 2007

19. Ihab Kazem, Dewan Tanvir Ahmed, Shervin Shirmohammadi, "A Zone Based Architecture for Massively Multiuser Simulations", Proc. SCS/ACM Communications and Networking Simulation Symposium, Norfolk, VA, USA, March 2007

20. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "Multi-Level Hashing for Peer-to-Peer System in Wireless Ad Hoc Environment", Proc. IEEE Workshop on Mobile Peer-to-Peer Computing, White Plains, NY, USA, March 2007

21. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "BM-ALM: An Application Layer Multicasting with Behavior Monitoring Approach", Proc. IEEE International Symposium on Multimedia, San Diego, CA, USA, December 2006

22. Dewan Tanvir Ahmed, Shervin Shirmohammadi, Jauvane C. Oliveira, "A Novel Method for Supporting Massively Multi-user Virtual Environments", Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, ON, Canada, November 2006

23. Dewan Tanvir Ahmed, Shervin Shirmohammadi, Ihab Kazem, "Zone Based Messaging in Collaborative Virtual Environments", Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, ON, Canada, November 2006

24. Dewan Tanvir Ahmed, Shervin Shirmohammadi, "A Hybrid P2P Protocol for Real-time Applications", 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, Manchester, UK, June 26-28, 2006

1.6 Road Map

The road map of the thesis is as follows: Chapter 2 illustrates the necessary background and relevant literature reviews for virtual collaboration architectures, ALM-based protocols, AoIM, and load balancing. Chapter 3 outlines a strategic model on how to choose an application-specific ALM protocol. The zonal hybrid MMOG architecture and its communication structure are outlined in Chapter 4. Several load-balancing algorithms for zonal MMOGs are presented in Chapter 5. In Chapter 6, several performance enhancement mechanisms such as dynamic area of interest management, expedited state dissemination, and quality improvement procedures are presented. The simulation results and validation of the presented model and algorithms are given in Chapters 7 and 8, respectively. Finally, the thesis is concluded in Chapter 9 with suggestions for future research.

Chapter 2

Background and Related Work

Since the introduction of the *networked virtual environment* (NVE) in the 1980s for military simulations, many interesting applications have evolved. The NVE and its variants are hot research topics that need to be explored. The genre of multi-player online games is relatively new but increasingly popular. The development of online games requires overcoming several technical challenges including consistency, responsiveness, reliability, and synchronization [Ale05][McF05]. For a consistent and reactive game space, each player maintains a clone of the appropriate game states in his station. When a player performs an action or generates an event that affects the game space, the game state of all other players influenced by that action or event must be updated. As consequence, the amount of information that must be exchanged among players roughly depends on game rules, density of players in an area, and game perspectives. However, capacity is bounded by at least two technical limitations: network bandwidth and processing power [SZ99][SKH01]. Much research has been conducted on the various aspects of networked virtual environments and MMOGs. This chapter will cover this research in detail and present a preliminary understanding of the system.

2.1 Zoning and Area of Interest

For easy state administration, the virtual space is divided into multiple adjacent areas, technically called zones. But the perspective from which a zone is constructed is subject to specific implementation. From a networking perspective, each LAN can be considered a zone where several LANs are connected through the Internet, forming the entire world. A LAN provides high bandwidth, so it could be easy for the server, i.e. the master, responsible for a zone to construct the overlay network if required, maintain its state, and manage newcomers and early departures. However, this is not a sufficient requirement; other factors such as virtual distance and participant's visibility scope need to be taken into consideration when defining a zone. Thus, a zone involves a logical partitioning, which is usually transparent to players in the game space.

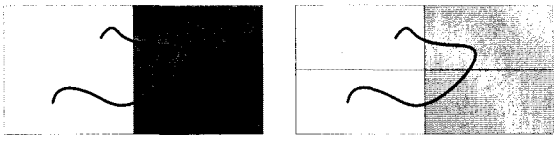

Figure 2.1: Two versus multiple zones layout, with a player moving across zones

2.1.1 Multiple Zones and Their Shapes

At its simplest, a zone can be represented by a square or a triangle. Multiple-zone definition can be adopted while defining the map of a game space. To accommodate many players, the map is logically divided into multiple zones. Each zone encompasses the players that are in the same vicinity. Henceforth, when a player moves from one zone to another, the player is disconnected from one server and joined to another server. If this multiple-zone layout is considered, more connections and disconnections

can be encountered for the same path traversal scenario (Figure 2.1). Triangular and hexagonal shapes have an advantage over circular shapes as these shapes can stick together and cover the entire game space (Figure 2.2).

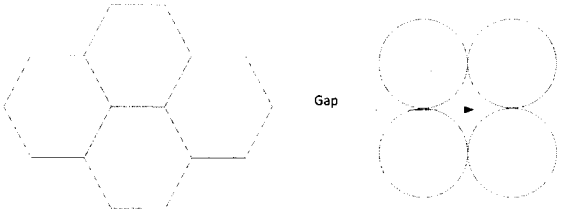

Figure 2.2: Hexagonal versus circular zone shapes

2.1.2 Area of Interest Management

An MMOG deals with plenty of information: monitoring each player's activities, tracking its location and many others. Rationally, a player does not need state information about the entire virtual world, which is too large. Thus, determining correct information for each player is a fundamental requirement of online games. From this perspective, interest management is a way of determining the functional details of a player. Thus, the performance of a virtual world depends on the cost and effectiveness of the AoIM approach deployed.

Publisher-Subscriber Model

Interest management for an MMOG can be abstracted using a publisher-subscriber model. The concept is that *publishers create events and subscribers consume events.* In this model, interest management consists of determining when an avatar registers to or bows out, gracefully or ungracefully, from a publisher's (avatar's) updates. Generally, interest management for online games can has multiple domains. The most common domain is visibility, although other domains like audible range and radar are

also possible. Each interest domain has special properties for the transmission and reception of data, so different sets of publisher-subscriber models might be needed.

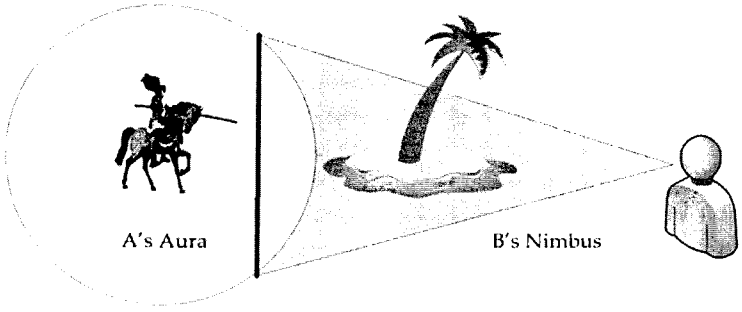

Figure 2.3: The aura-nimbus model

Space-based AoIM Model

Interest management can also be categorized into two general groups: space-based and class-based. Space-based interest management can be defined based on the relative position of avatars in a virtual world, while class-based can be determined from an avatar's attributes. Space-based interest management is the most useful for MMOGs because of the relevant information of a player is usually closely related to its position in the environment and is typically based on proximity, which can be realized in terms of the *aura-nimbus* information model illustrated in Figure 2.3. *Aura* is the area that bounds the existence of an avatar in space, while *nimbus*, i.e. area of interest, is the space in which an object can perceive other objects. In the simplest form, both aura and nimbus are usually represented by circles around the avatar. This model is more appropriate when a server maintains a connection with each client. The drawback of a pure aura-nimbus model is scalability, because of the computing cost associated with the determination of intersection between nimbus and aura for a large number of players.

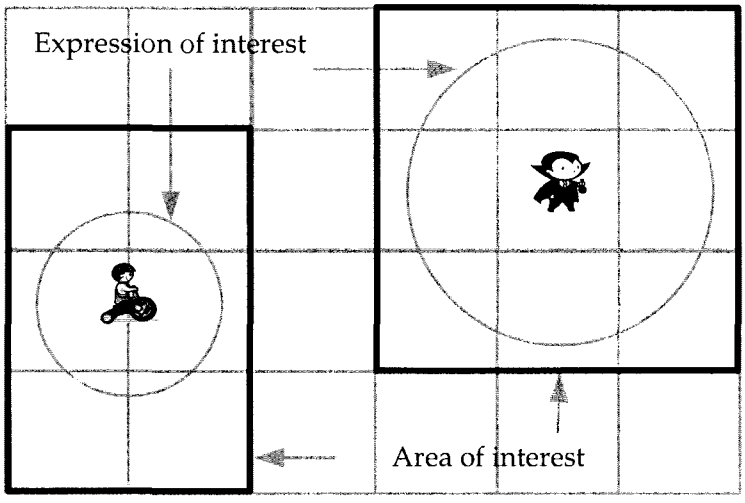

Figure 2.4: Region-based area of interest

Region-based AoIM Model

Region-based interest management partitions the game space into several fixed regions. The interest management scheme then determines which regions intersect with the players' expression of interest. Thus, an area of interest becomes a union of intersecting regions with respect to the expression of interest, as shown in Figure 2.4. This is an approximation of true expression-of-interest and generally cheaper to compute but less precise than a pure aura-nimbus model.

In reality, a significant portion of a large game space is irrelevant to a player. It is apparent that the simulation data space should be filtered based on relevance and channeled efficiently to the appropriate players. So at a particular instant in time, such 'relevant' information depends on a player's place and its surroundings, which are influenced by metrics like virtual proximity, field of vision, line of sight, action context, etc. Events should be sieved in a communication model according to players' interest, which could reduce stress on latency and congestion [ERMS06].

2.1.3 Discrete View versus Continuous View

An MMOG can have a *discrete view* for its players where a player is interested in a single zone. The idea is covered in P2P Support for Massively Multiplayer Games [KLXH04] and Zoned Federation of Game Servers [IHK04]. The discrete view simplifies the design and makes it more scalable and robust. However, in reality, simulation and game mechanics require continuous view, where nodes can see across zones. Thus, the area of interest management cannot assume a discrete view for all entities and must support continuous view for frequently occurring events and for special entities like radar. Boulanger et al. present a nice comparative study of different interest management algorithms [BKV06]. Eight different approaches are compared and evaluated in the context of MMOGs. One of the interesting regulations is that an interest management algorithm that considers obstacles in a virtual world reduces update messages exchanged between players.

2.1.4 Geographic versus Behavioral Modeling

There are two approaches to model AoIM for MMOGs. The first is *static geographical partitioning*, implemented at the initialization phase of a simulation. This is practical as it describes the structure of a virtual world. For example, a virtual world may consist of multiple cities where each city defines a geographical partition: it is the area where most of the interactions take place, and in most cases, participants are not affected by what is happening in other cities. *Second Life* has adopted such an approach [VBV+05]. The virtual world can place uninteresting items around the borders, such as cities separated by empty forests or wilderness where players do not want to stay long. Although static geographic partitioning is good for some cases, it might not be a general solution for all virtual simulators.

The second approach for AoIM is *behavioral modeling.* In military simulations, two different units, such as a jeep and an aircraft, have different characteristics in

terms of how fast they can move, how far they can see, and the scope of their interaction space (a jet launching a missile has a larger area of influence than a jeep on patrol). Lu et al. argue that, as the mapping of processing resources to geographic regionalization is straightforward and uncomplicated, the behavioral approach has not been deeply explored [LPM06]. One of the limitations of a geographic regionalization is its unintelligence in preventing inter-server communications. This is because the geographic regionalization does not give enough importance to players' interactions. Even though behavioral modeling is the ideal approach to manage interest among parties, geographic regionalization is not without its merits. Thus, geographic regionalization can be augmented by behavior-based communications for better interest management.

2.2 Networking and Scalability for MMOGs

Since the introduction of NVEs, several architectures have appeared, with a wide variety of approaches and characteristics. Hu et al. propose, in some sense, a fully distributed peer-to-peer architecture to solve the scalability problem of networked virtual environments [HCC06]. This method exploits the locality of player's interest inherent to the system and is based on the mathematical construct known as a Voronoi diagram. In this concept, the game space is dynamically partitioned depending on players' position using a Voronoi graph-partitioning algorithm. Thus, players in the same region can directly exchange game events and maintain a consistent game space. However, as the number of player nodes in a sub-state increases, the number of messages sent from each player or its management node also increases. Thus, communication and message exchange cannot be regulated properly. Here, the position of players is managed in a centralized way that creates a big problem for scalability.

Marios et al. present an approach to support massively multiplayer online role-playing games (MMORPGs) using a centralized distributed architecture [AT06]. This

approach considers the player's locality of interest to reduce bandwidth requirements for both game servers and clients. But from an architectural point of view, it is simply a multiple server-based client-server architecture where performance improvement is flat. There is no guarantee of end-to-end delay. Here, a player state includes a set of all other players and servers that currently know this player. However, if a player leaves, it is not clear what will happen to others, i.e. how will this departure be handled? This effect of player departure is not addressed in this architecture.

The model proposed by Hampel et al. reuses architectures capable of exploiting the flexibility and scalability of peer-to-peer networks [HBH06]. The main drawback of peer-to-peer networks for games is the lack of a central authority that can regulate access and prevent cheating. This model uses a set of controller peers that can supervise each other. This kind of redundancy can prevent cheating. The model is based on the existing distributed hash table Pastry, which has been extended into SCRIBE. The key issue is unbounded end-to-end delay, which could be a problem for synchronization.

Yamamoto et al. present a load-balancing mechanism for a crowded sub-space [YMYI05]. The proposed technique reduces end-to-end event delivery delay through a load-balancing tree by replacing one of the intermediate nodes with the backup node incrementally. This also presents a technique for efficient and seamless sub-space switching for subscription while each player's view can move in the game space. For each sub-space, a player node called the responsible node is selected. The responsible node forwards events to all players in the same sub-space, but the technique as described does not explain on what basis such nodes can be chosen or consider the ability of nodes to perform such a critical task. In short, it is clear from the description that event distribution is performed from one-to-all nodes in a sub-space. This pattern resembles client-server architecture, which is a big concern for scalability.

Knutsson et al. describe peer-to-peer support for massively multiplayer games by using Pastry and SCRIBE, a peer-to-peer overlay and its associated simulated

multicast structure [KLXH04]. The virtual world is divided into regions of fixed size. Each region is managed by a coordinator, the root of the multicast tree. Players inside the same region subscribe to the address of the root node to receive updates from other players. Thus, neighbors are discovered via the coordinators. The coordinators maintain links with each other, easing player transition to other regions. However, this model has some undesirable properties. Due to discrete AoI, users cannot see across regions. If players decide to listen to more regions, as suggested in the paper, unnecessary messages beyond AoI will be received. This can create a serious performance penalty as the overlay does not cover the appropriate area of interest; messages may need to be relayed by other nodes (one to two hops in most cases, but some cases may go beyond 50 "virtual hops", so more delays can happen at the physical level). In short, the architecture does not fully use the power of direct connections. There are many key differences between MM-VISA, presented in this thesis, and their approach. One of the salient characteristics of MM-VISA is its continuous view for the player, even with multiple zone layouts. MM-VISA has a sound zone switching mechanism as well as several intelligent techniques to avoid frequent connections and disconnections with zone masters.

The *Delaunay network* is a good solution for NVEs in that it organizes players according to their positions in the virtual space [VBD07]. The maintenance cost of a Delaunay network increases against players' density and velocity. Thus, a player may generate a considerable volume of traffic to be dealt with. To address this issue, authors propose a dynamic clustering algorithm where each peer in the network monitors its cost of maintenance and creates a new cluster when the volume of traffic exceeds a given threshold. The members of a cluster then expand their coordinates to increase their reciprocal distances. In this way, by decreasing the concentration of players, the system tries to reduce maintenance cost. But its centralized architecture is the main drawback for scalability. The architecture of MM-VISA is an integration of centralized servers and distributed peers that retains the benefits of both systems.

It achieves scalability through the use of peer resources while maintaining the central power of the system in the different aspects of group management.

SCORE (*scalable multicast-based communication protocol*) is designed for *Large-Scale Virtual Environments* (LSVE) over the Internet [LTB04]. To handle a large number of participants, it supports multiple multicast groups and multiple agents. It dynamically partitions the virtual world into spatial areas and applies planar point processes to determine a proper cell size. Thus, it ensures traffic at the receiver side below a threshold with a given probability. Although the goals of SCORE and MM-VISA are similar, i.e. large-scale collaboration, MM-VISA focuses specifically on networked games. Hence, as the application domain is different, every aspect of the design varies, from zone definition to both intra- and inter-zonal communications.

MOPAR, a peer-to-peer networked game architecture, is a scheme for interest management in NVEs [YV05]. It is a combination of both structured and unstructured peer-to-peer systems. Here, a master node is chosen in each zone and becomes the parent of all other players, named slaves, in the zone. Each master node supports all slaves within its zone. Although the architecture is peer-to-peer in the sense that the master node is also connected to other master nodes and manages inter-zonal communication, the networked architecture within a zone looks like the client-server architecture. Thus, it has a single point-of-failure problem. One of the main drawbacks of MOPAR is unexploited slaves' bandwidth as slaves are only connected to the master node, not among themselves. On the other hand, MM-VISA offers a scalable system that can reduce the master's workload through peers' participation.

A. Steed et al. propose a simple but powerful visibility structure called frontier sets [SA05]. The proposal shows how to construct this set at runtime. For a pair of nodes, a frontier identifies two mutually invisible regions containing nodes. The frontier set allows a system to scale. This is possible because, as long as two nodes stay in their respective frontiers, they do not need to send update information to each other. This is an interesting method in theory. However, it would be computationally

expensive to realize it in real-time and, in fact, would be difficult, especially when offering continuous views to the players. The MM-VISA can take care of these issues through the use of multiple multicast groups and can maintain a low data rate through locality awareness.

The key issues in CVE research include managing consistency and persistency of distributed information and assuring real-time interactivity. Fook et al. present *Collaborative Interaction Management* (CIM) and *Task-Oriented Interaction Management* (TIM) approaches to resolve extensibility issues in CVE [FQL03]. When multiple interactions occurred at the same time, these approaches can govern and control the message flow. Here *Scene Interaction Manager* (SIM) monitors the network characteristics to prevent load saturation and large network delay. This approach is mainly conceptual that does not consider the key features of real MMOGs like interest management. From this perspective, it is not particularly realistic and not very well-suited for MMOGs.

ATLAS focuses two broad concepts to support users in collaboration in heterogeneous environments [LLH02]. It goes for self-tune-ability rather than for reconfigurability, where a system automatically configures itself based on the current execution environment. The other concept associated with ATLAS is personalized information filtering. Based on human heuristics, application semantics, user preferences and current system status, it filters out events to increase scalability without degrading interactive details for the user. A virtual environment allows users on a network to interact with each other by sharing the common view of their states. ATLAS analyzes scalability in terms of communication architecture, interest management, concurrent control, and data replication.

To support a large number of concurrent participants, Z. Liang proposes a mobile agent-based architecture for a *Large-scale Collaborative Virtual Environment* (LCVE) [LQF03]. This software system is made up of mobile agents, and each mobile agent is responsible for different independent tasks in the LCVE. Theoktisto and Fairn

propose a component framework for transforming standalone virtual reality (VR) applications into multithreaded collaborative virtual reality environments [TF05]. Their approach includes a hybrid distributed user interaction model, multithreaded software components and network communications under a peer-to-peer scalable topology.

The accessibility of fast Internet connections and the availability of cheap and smart graphics cards have made networked virtual environments viable for millions of users. Now it is time to cope with the growing heterogeneity that arises from the differences in computing power, network bandwidth and users' preferences. H. Trefftz et al. present a mathematical model to integrate this heterogeneity that considers policies and users' preferences as the controlling parameters in a linear equation [TMZ03].

2.3 Application-Layer Multicasting

Unlike IP multicast, where the routers take care of routing and avoid multiple copies of a packet over the same link, possibly constructing optimal trees, ALM is implemented by application nodes (either end-systems or proxies) and results in multiple copies of a packet over the same link as well as typically constructing non-optimal trees. In exchange for its inefficiency, as compared to IP Multicast (higher-stress links and larger-diameter trees), ALM remedies the key shortcomings of the IP Multicast model: it promises easier and possibly immediate deployment over the Internet. End System Multicast (ESM), one of the current implementations of ALM, has already been deployed successfully on the Internet for various applications. As online games require group communications, ALM can be a promising alternative to native IP multicast considering its limitations. There have been significant research activities in ALM-based protocols. We discuss some of the well-known approaches, mostly related to collaborative applications. The classification of different ALM protocols both in terms of group management and routing policy can be found in [HASG07].

Kim et al. propose an approach that constructs topologically-aware data paths, which are based on topological clustering of multicast group members [KC04]. This approach hierarchically arranges the clusters and separates data paths into two types (i.e., *inside-cluster path and outside-cluster path*) to exclude outsider nodes from the inside-cluster paths. Topologically-aware data paths can reduce unnecessary high latency and redundant network resource usage with a low overhead.

Wierzbicki et al. introduce *Fastcast* for efficient peer-to-peer applications [WSB03]. This is a root-based topology-aware ALM protocol. This ALM algorithm can adjust performance by limiting the number of children in an ALM tree, since nodes could be connected by slow modems or slow wireless connections. The MM-VISA considers resource limitations of end-hosts and assigns out-degree/fan-out accordingly as a preventive approach against device heterogeneity.

*Yoid*is an ALM protocol that constructs a multicast tree using distributed end-hosts [Fra99]. It uses hop count as a measure of distance. Guo et al. propose a solution to deal with the problem of disruption in live video streaming for a group of clients [GA04]. Video continuity is maintained in spite of the departing clients using a combination of time-shifted streams and video patching. Both techniques need high bandwidth not only from the server but also from the end-users. Thus, neither is readily deployable for virtual collaborations.

Brosh et al. propose a new model that directly maps node load to the delay penalty at the application hosts [BLS07]. The trade-off is either to select shortest path trees or to restrict load on the hosts. Another interesting application layer multicast protocol is the VRing (*Virtual Ring*) [SYH04]. This protocol establishes a virtual ring among the multicast group members in a self-organizing and distributed manner that provides less control overhead, consumes less bandwidth, and provides lower average node degree at the cost of higher path stretch and higher link stress than some other ALM protocols. The main problem of a ring-based topology is the large routing delay.

NICE is a recursive acronym that stands for *Internet Cooperative Environment* [BBK02]. This scalable application layer multicast protocol uses a hierarchical clustering approach to support a larger number of receivers. NICE was designed to provide architectures for low-bandwidth soft real-time data-streaming applications such as real-time stock quotes and updates, and Internet radio. It organizes hosts in hierarchical layers where each layer has several clusters of hosts. The lowest layer in the hierarchy is denoted by L_0. The size of a cluster is between K to $3K-1$, where K is a constant. Each cluster has a leader to communicate with the higher layer, which is chosen at the center of the cluster. Thus, the leader has the minimum maximum distance to all other hosts in the cluster. NICE is different from other protocols in the sense that its *data delivery* paths form a simple loosely-connected tree (loop-free structure) while *control paths* are a clique (a strongly connected structure to reduce traffic and to quickly detect changes and restore invariants).

The *Minimum Spanning Tree* (MST) and *Shortest Path Tree* (SPT) routing algorithms can be modified to comply with the degree constraint of each node. The problem of finding minimum-cost degree-constraint multicast trees or degree-constraint Steiner trees is NP-complete [Dou92]. There exist several heuristic approximation algorithms addressing this problem [CLR93][KR00][KR03]. Some of these algorithms (such as [KR00][KR03]) cannot provide exact guarantees on the degree of each node in the tree and instead provide a bound on the worst-case degree. Others focus on constructing a single tree and do not consider multiple trees over the same graph ([Dou92] [ST02]). Though there has been some research constructing multiple trees on a shared graph [CGY00], they still only provide a bound on the worst-case (maximum) degree of any node as opposed to guarantees on the individual maximum degree for every node which is required for a protocol supporting multi-source applications. Designing a protocol typically involves making the design decisions based on a given set of requirements, and most importantly respecting the constraints in all circumstances. For intra-zonal communication, MM-VISA considers a new ALM protocol

that is robust even in peer dynamics. The zone master constructs a mesh through a greedy heuristic method subject to degree constraints. As a result, the constructed mesh covers all players (i.e. peers) with a lower stretch and provides a good platform for multi-source collaborations. The routing tables are constructed through the dominating set principles to address the ad hoc nature of the players.

2.4 Load Balancing

Traditionally, each user in an NVE is only interested in a small portion of the world where most of the interactions take place. This interaction space is called the user's *Area of Interest*(AoI). In an MMOG, it is possible that many players can move into a zone, degrading gaming quality and affecting players' gaming experience. This problem is known as the *hotspot problem*. Chen et al. mention that flocking (i.e. the appearance of a hotspot or too many players moving into a zone administered by one server) is an MMOG pattern that cannot be avoided [CWD+05]. This is natural as some areas in a game are more interesting than others. Moreover, hotspot development is unpredictable as it depends on game events like players chasing each other or performing a mission. Chen et al. also mention that in most cases, static partitioning handles it poorly [CWD+05].

According to Duong et al., dynamic load distribution methods can be categorized into load-sharing algorithms and load-balancing algorithms [DZ03]. Load-sharing algorithms prevent unbalanced load among servers. Load-balancing algorithms address the situation by equalizing the workload of the servers in a distributed manner. The computational overhead associated with load balancing impedes an algorithm's effectiveness, and subsequently the gaming experience. So, this fact cannot be ignored.

A preventive mechanism for migration is to impose boundaries between zones: cities are separated by forests or wilderness, where parties cannot stay long [VBV+05]. Thus, a migrating player cannot experience others' interactions at boundaries. But

such restrictions cannot always be imposed, and even if applied, they cannot avoid hotpots in "battle" situations where everyone must be in the same zone facing the enemy. Vleeschawer et al. introduce a microcell approach to balance load [VBV+05]. Instead of only managing a zone or a microcell, each server is responsible for a cluster of microcells. Different algorithms are presented and tested to show how microcells can be clustered to reflect players' distribution within a map. In this way, an optimal configuration can be determined to balance load. However, this clustering is only applicable when the game is first loaded. Microcell clustering needs to be done in real time, but it is expensive and impractical to run such algorithms dynamically in the simulation at a full phase. Needless to say, to reach an optimal mapping of clusters-to-servers, one should assume global knowledge of load distribution and events in the map, which is inappropriate in distributed environments [CWD+05].

Lu et al. also present some techniques for load balancing [LPM06]. An ideal solution ensures equal load distribution to avoid exhausting one server while keeping spare resources at other servers. Theoretically, this is an ideal solution and does not interrupt players' gaming experience. But assessing load on servers, frequent migration of players from one sever to another, and tracking areas of interest are big problems in this situation. It would be even more difficult for hybrid systems where players have routing responsibility.

Chapter 3

Application-Layer Multicasting and P2P Communications

In light of the slow deployment of IP Multicast technology on the global Internet and the tremendous popularity of peer-to-peer file-sharing applications, there has been a flurry of research investigating the feasibility of implementing multicasting capability at the application layer, referred to as *Application Layer Multicasting* (ALM), and numerous algorithms and protocols have been proposed. This chapter provides an understanding of ALM protocols by identifying significant characteristics, both from application requirements and networking points of view, and by using these characteristics as a basis for organizing the protocols into an integrated and well-structured format.

3.1 Trade-off: ALM vs. IP Multicast

The concept of ALM is simply the implementation of multicasting functionality as an application service instead of a network service. Figure 3.1a represents an ALM configuration for the same group of senders and receivers in the IP multicasting

scenario shown in Figure 3.1b. Here, the multicasting tree has been built at the application layer. Using only the unicast capability of the network, the source sends two packets, one to D1 and one to D2, each of which in turn sends the packet to D4 and D3, respectively. While IP Multicast is implemented by network nodes (i.e. routers) and avoids multiple copies of the same packet on the same link as well as possibly constructing optimal trees, ALM is implemented by application nodes (either end systems or proxies) and results in multiple copies of the same packet on the same link as well as typically constructing non-optimal trees. In exchange for its inefficiency, as compared to IP Multicast (by resulting in higher-stress links and larger-diameter trees), ALM remedies the key shortcoming of the IP Multicast model: it promises easier and possibly immediate deployment over the Wide Area Network. For example, End System Multicast (ESM) [1] [CRZ02], one of the current implementations of ALM, has been already deployed successfully on the Internet in various applications. In ESM, when a user tunes into the system, this end-host can download data and upload it to other end-hosts.

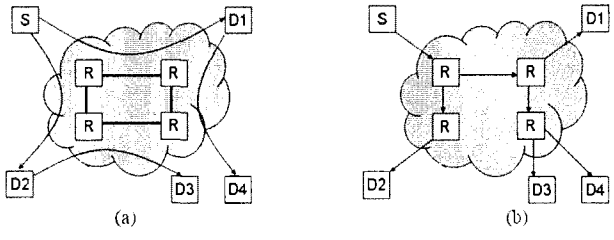

Figure 3.1: (a) Application layer multicast (b) IP multicasting scenario

Chu et al. illustrate, using both simulation and Internet experiments, that ALM systems can form overlay multicast trees that introduce low performance penalties (in terms of link stress and tree stretch) compared to IP Multicast [CRZ02]. ALM disadvantages in comparison to IP multicasting, such as longer delays and less efficient

[1] http://esm.cs.cmu.edu/technology

network usage, are balanced by its advantages, such as immediate deployability on the Internet, easier maintenance and updating of the algorithm, and last but certainly not least, the ability to adapt to a specific application. A common approach to Application Layer Multicasting for the multicast participants is to establish an overlay topology of unicast links to serve as a virtual network (overlay network) on top of which multicast trees can be constructed. Figure 3.2 shows an example of seven peers forming a topology (Figure 3.2a) and a multicast tree being constructed with node D as the source (Figure 3.2b).

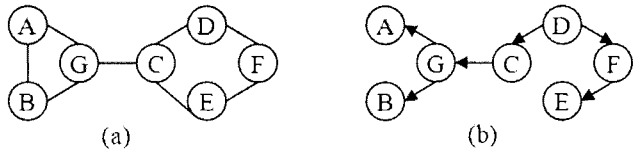

Figure 3.2: (a) Sample overlay topology (b) an overlay multicast tree

To better illustrate the performance penalties mentioned above, let's take a closer look at a scenario comparing IP Multicast and ALM. Consider Figure 3.3a, which shows a physical topology. There are four routers (A-D), and four end-systems (a-d). Link delays are as indicated. Assume 'a' wishes to send data to all other end-systems. Figure 3.3b depicts the IP Multicast tree constructed by *Distance Vector Multicast Routing Protocol* (DVMRP). Routers A and C receive a single copy of the packet and forward it along multiple interfaces. At most, one copy of a packet is sent over any physical link. Each recipient receives data with the same delay, as though end-system 'a' were sending it directly by unicast. On the other hand, ALM does not rely on router support for multicast. Here, data replication and forwarding are handled by the end-systems, as shown in Figure 3.3c. Figure 3.3d shows how the end-system overlay network maps onto the underlying physical network. The resource usages of IP multicast and ALM for this particular case are 37 and 39 respectively. ALM

is therefore more "costly" in this example, again balanced with the benefit of being immediately deployable.

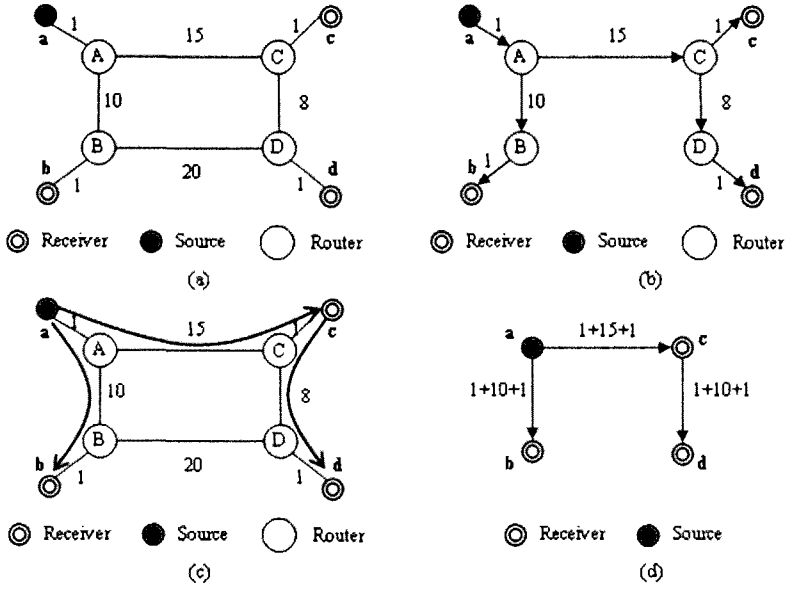

Figure 3.3: (a) A physical topology (b) IP multicast tree constructed by DVMRP (c) ALM concept (d) End-system overlay network

Multicast routing protocols build multicast trees to deliver data and to exchange necessary routing information. In IP multicast, each host informs its designated multicast router in its sub-network when it joins or leaves the group. Then the multicast routers exchange group membership information over the multicast tree. All of this control overhead about members joining, members leaving, and updating the multicast tree is carried by the *Internet Group Membership Protocol* (IGMP) [Fen97]. As there is no redundant path in the tree, IP multicast improves network efficiency and scales to a large group size. Despite its bandwidth efficiency, it suffers from the deployment issues mentioned earlier. Application layer multicast, although less efficient than IP Multicast as demonstrated in the above comparison, is becoming increasingly

Table 3.1: Conceptual comparison of IP Multicast and ALM

Issues	**IP multicast**	**ALM**
Multicast efficiency in terms of delay/bandwidth	High	Low - Medium
Complexity or overhead	Low	Medium - High
Ease of deployment	Low	Medium - High
OSI layer where the multicast protocol works	Network layer	Application layer

popular in the multicast community primarily due to its ease of deployment. In ALM, multicast architecture, group membership, multicast delivery structure construction, and data forwarding are exclusively controlled by participating end-hosts; thus, it does not require the support of intermediate nodes such as routers. Negatively, end-hosts in ALM have little or no knowledge about the underlying network topology, thus resulting in performance penalty in terms of longer latency and lower efficiency compared to IP multicast. Group membership and multicast delivery structures and monitoring of network conditions are also performed at end-hosts, causing additional overhead for end-hosts compared to IP multicasting. Table 3.1 is a conceptual comparison of typical IP multicast and ALM.

In ALM, new members learn about the topology from a common bootstrap point called a *Rendezvous Point* (RP) and join the topology by exchanging control messages with a subset of members already part of the topology. Unlike the IGMP protocol used in IP Multicasting, the control messages in ALM are exchanged in an application-specific manner and are completely up to the designers of the protocol. A "good" topology consists of a rich connected graph, such that a peer is connected to other peers through multiple paths, and in an efficient and cost-aware manner, such that

the distance or delay between peers is minimized while the number of connections is bounded. Other metrics such as robustness (ability to deal with members leaving the topology), scalability (ability to efficiently increase the size of the topologies for a very large number of peers) and low control overhead (minimizing the exchange of control messages) also indicate the quality of an overlay topology. Creating and maintaining 'good' topologies thus becomes one of the core responsibilities of ALM protocols. Once a topology is constructed and maintained, a multicast tree can be constructed on top of the graph according to a routing strategy that would commonly strive to minimize the cost of the multicast tree in terms of the delay (or other important parameters, depending on the application) experienced by each peer as well as the amount of data duplication required with each peer. Revisiting Figure 3.2, we can say that Figure 3.2b shows an example of overlay tree over the sample topology of Figure 3.2a. In the next section, we will take a closer look at these design issues related to ALM topology and multicast tree.

3.2 ALM Protocol Design

Since its introduction, there have been many ALM protocols with a wide variety of approaches and characteristics. Designing a protocol typically involves making design decisions based on a given set of requirements, constraints under certain circumstances and a given set of resources whose availability is assumed. The aim of this section is to highlight some of the important categories and general approaches of different protocols based on these requirements, constraints and resource assumptions and discuss how they affect the service each protocol provides as well as its overall characteristics.

3.2.1 Application Domain

Perhaps the most crucial feature of an ALM protocol and one that affects most of its resulting characteristics is its targeting application. The application domain determines the number of users that a protocol must support, the data types a protocol's delivery tree must accommodate and the metrics that such a tree attempts to optimize. We follow the same categorization of application domains driving multicast deployment as those according to Diot et al. [DLL+00]:

1. *Audio/video streaming*: usually involves a single source distributing media to a large number of receivers. Examples include live streaming of a sporting event, or streaming of pre-recorded news. The primary metric is bandwidth and latency to a lesser extent.

2. *Audio/video conferencing*: these involve small to medium-sized groups interacting in a multi-party conferencing session. The difference from the previous category is the smaller group size, higher degree of interactivity and the existence of multiple sources. Both bandwidth and latency are important metrics.

3. *Generic multicast service*: protocols falling into this application domain try to create a generic multicast service based on specific metrics that can affect a variety of applications.

4. *Reliable data broadcast and file transfer*: reliable transfer and distribution of (usually large) files (e.g. distributed databases and file sharing). Bandwidth is the only metric.

As can be seen, the different classes of applications have different sets of requirements regarding reliability, latency, bandwidth, and scaling. Such requirements in turn determine the design choices of ALM protocol regarding the group management mechanism it deploys. The application domain therefore influences the ALM

protocol. In a tree-based multicast system, for example, a node is either an interior node (has children) or a leaf node (has no children). This design choice initiates two problems. First, it is an unfair system as only the interior nodes are responsible to forward the data. The system becomes unbalanced as leaf nodes increase more rapidly than the interior nodes. Second, due to network capacity, interior nodes may not handle high-bandwidth applications - sacrificing the quality. In an application-level streaming system, audio/video streams are usually split into several smaller streams. Each stream is stamped with a numerical sequence number to place it in the correct sequence for playback. Usually FEC (forward error correction) code is used to ensure guaranteed stream delivery. For example, Split stream [CDK+03] ensures that the majority of nodes are interior nodes in one tree, and they will be leaf nodes in all other trees. Hence, the system distributes forwarding workloads among all nodes and solves the unfair-and-unbalanced problem in the conventional streaming system. In Split stream, nodes choose to join a subset of the stripes to control their inbound bandwidths and also opt to limit the number of children nodes they accept to control their outbound bandwidths. Thus, it accommodates nodes with different bandwidths and solves the second problem. Similarly, other application domains could have different objectives and different constraints. Typically, an ALM protocol focuses on optimizing its tree designed for a specific application domain.

3.2.2 Deployment Level

A key factor determining the set of assumptions on whose basis an ALM protocol operates is at what level the protocol is expected to be deployed: at the infrastructure level or end-system level. Infrastructure-level, also known as proxy-based ALM protocols, requires the deployment of dedicated servers/proxies on the Internet, where they self-organize into an overlay network and provide a transparent multicast service to the end-user (Figure 3.4a). End-system-level ALM protocols, on the other

hand, assume only a unicast service from the infrastructure and expect end-system hosts to participate in providing multicasting functionality by taking on some of the forwarding responsibility (Figure 3.4b). Figure 3.4 highlights the difference between the two approaches to ALM.

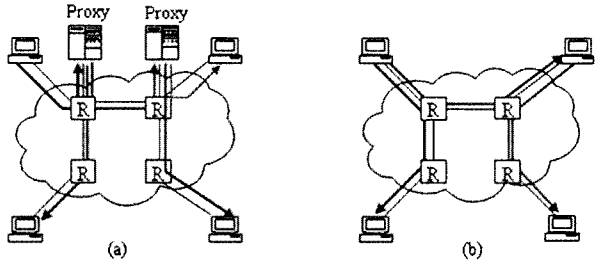

Figure 3.4: (a) Proxy-based deployment of ALM (b) End-system ALM

The choice between developing an infrastructure level or an end-system-level ALM protocol is perhaps driven as much by business and marketing issues as by purely technological ones. End systems sharing the forwarding load of a multicast session use the existing Internet infrastructure available to them and may not be expected to pay more for participating in the multicast session (as illustrated by the free nature of peer-to-peer file-transfer applications). An infrastructure of dedicated proxies deployed over the Internet that offer multicasting services however are more likely to expect a service charge. There are, however, technological consequences for a choice between a proxy-based and an end-system-level approach to ALM.

Proxy-based ALM protocols can take advantage of existing IP Multicast 'islands' by including a representative of an island as an overlay node (and therefore increase their efficiency), assume greater bandwidth availability to the proxy nodes (compared to the bandwidth available to end-systems), assume the longer life cycle of overlay nodes (compared to the transient nature of end systems), relieve end-systems from any forwarding responsibility, and therefore reduce application complexity since mul-

ticast is transparently made available to end-systems. The major disadvantage of this approach is the need for the deployment of dedicated proxies over the inter-network, thus incurring the cost associated with acquiring and deploying them. Proxy-based ALM may also be less adaptable to and less optimized for applications since it would typically provide a generic multicast service rather than a service specific to a particular class of applications.

End-system ALM protocols have more flexibility and adaptability to specific application domains and immediate deployment over the Internet, but they may not scale well (either to a large number of users or to a large number of simultaneous sessions), must deal with the limited bandwidth of end-systems and require end-systems to take some of the forwarding responsibilities (and therefore increase application software complexity).

3.2.3 Group Management

Once application domain and deployment level has been decided, a protocol designer must make some key decisions regarding how to manage a group of nodes in a multicast session. This includes:

1. Basic group management: how users find out about multicast sessions, how they join a session (whether through a rendezvous point, or if p2p substrate is required and some form of flooding is used to find the appropriate source), how they leave (depending on how permanent and cooperative the users are assumed to be), and whether the users still contribute to existing multicast session even if they are not a part of them. Are they assumed to be very transient and anonymous or more permanent and known users?

2. Whether the management of the group is done in a centralized or distributed way and how this affects the design and service provided.

3. Whether to take a mesh-first approach or a tree-first approach. What are the advantages and disadvantages of each? If a mesh-first approach is chosen, is a peer-to-peer substrate is assumed to exist, and if so, what type of substrate with what requirements and services does it provide?

4. Whether the protocol will take advantage of existing IP Multicast islands in order to reduce part of the multicasting load. If so, how will the protocol interface to these islands?

5. Depending on the assumed lifetime of the multicast sessions, whether it is necessary to refine the multicast tree to improve performance as well as deal with fluctuations in the network resources available and deal with congestion. If so, the designer must also determine how aggressive these refinement methodologies can be and their effects on the stability of the system and the service provided to users.

The basic group management services that an ALM protocol can provide consist of a mechanism for the new nodes to discover a multicast session (typically through rendezvous point(s)), a distributed or centralized administration, and the mesh-first or the tree-first approach for constructing source-specific or shared trees based on some metrics. Such characteristics of group management mechanisms are primarily driven by the application domain. For instance, single-source video streaming with a large number of receivers usually involves a distributed group management and the construction of a source-specific tree based on bandwidth and delay metrics, whereas medium-sized conferencing applications may involve the mesh-first construction of a shared tree based on bandwidth and delay and can afford a centralized approach to group management. These characteristics are described next.

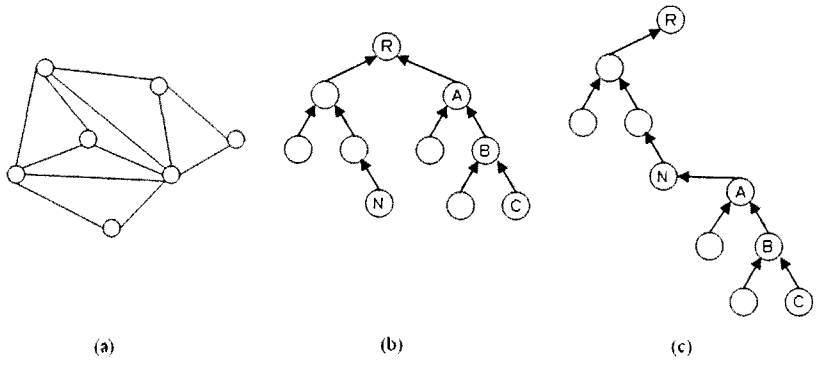

Figure 3.5: (a) A mesh: a network topology with many redundant interconnections between network nodes (b) Initial tree (c) Lopsided Tree

Mesh-First versus Tree-First

There are two basic approaches to configuring the data distribution pathways: mesh-first and tree-first. In the mesh-first approach, members keep a connected mesh topology (Figure 3.5a) among themselves. Usually the source is chosen as a root and a routing algorithm is run over the mesh relative to the root to build the tree. This mesh topology is explicitly created at the beginning; hence, it is known. On the other hand, the resulting tree topology is unknown. So the quality of the tree depends on the quality of the mesh chosen. By contrast, in a tree-first approach, the tree is built directly without any mesh. The members explicitly select their parent from the known members in the tree. This may require running an algorithm to detect and avoid loops, and to ensure that the structure is indeed a tree. There is no intervening mesh topology here. The reason for using the tree-first approach over the mesh-first approach is that the tree-first approach gives direct control of the tree. This control is valuable for different aspects such as maintaining strict control over the fan-out, selecting the best parent neighbor that has enough resources, or responding to failed members while minimizing impact on the tree. Another advantage of the tree-first

approach is independent actions from each member. It makes the protocol simple, as it has a lower communication overhead. But when a member changes a parent, it drags all of its descendants with it (Figure 3.5c). This is desirable in the sense that the descendants do not need to change their neighbors; in fact, they are unaware of the incident. However, this can also result in lopsided trees, which are "uneven" and less efficient than correctly formed trees. The advantage of the mesh-first approach becomes apparent here as it gives more freedom to refine the tree. It is possible to manipulate the tree topology to a significant extent by selecting mesh neighbors and changing the metrics. A mesh-first approach is therefore more robust and responsive to tree partitions and is more suitable for multi-source applications, at the cost of higher control overhead.

Source-Specific Tree versus Shared Tree

In multicasting, two conflicting design goals are (a) minimizing the length of the path (usually in terms of hops or end-to-end delay) to a specific individual destination and (b) minimizing the total number of hops or the cumulative end-to-end delay to forward the packet to all the destinations. To the best of our knowledge, there is as yet no good heuristic to balance these two conflicting goals. The choice between a source-specific tree (case a) and shared tree (case b) usually depends on whether or not the multiple sources use the same overlay for data distribution. Shared trees are preferred when there is a multiparty communication; i.e. multiple sources, such as online games. It is better than a source-specific tree in terms of maintenance cost. A source-specific tree, on the other hand, allows for optimization of the tree for a given source, but cannot efficiently support multiple sources on that tree.

Distributed versus Centralized

Although it might seem intuitive that a distributed routing approach would better suit large-scale applications to efficiently manage group communications, there are still incentives for a centralized approach [PWCS02]. In a distributed approach, the workload of maintaining the tree is evenly distributed among the root nodes. But the synchronous communication among the members for real-time applications like media streaming is hard to ensure due to the inherent decision-making delay in distributed techniques. The centralized management of multicast groups is a fair choice for small-scale applications. It is simple and easy to deploy. Naturally, there is always the risk of a single point of failure in a centralized system. Designers must balance simplicity and practicality against robustness when choosing one of these approaches in designing an ALM protocol.

IP Multicast Compatibility

It would be beneficial if an ALM protocol could exploit IP multicasting where it is available. This is advantageous for applications where the existing infrastructure of IP Multicasting (typically in a large organization or company) can be further enhanced to support Internet users. An example is the Hybrid Distributed Simulation Protocol (HDSP), which allows military simulations, traditionally performed on expensive networking infrastructure, to be extended to home users and/or between multiple multicast sites [DSG+06]. Another example is Island Multicast (IM), which integrates IP Multicast with ALM [LLS+05]. It has a two-level architecture, with the top level concerned with packet delivery between "islands" using a unicast mechanism and the bottom level concerned with packet delivery among the members in an island using IP multicast.

Refinement

Depending upon the order of joining requests for the same set of nodes, constructed trees could be different and have different perception qualities. The quality of an ALM path between any pair of members is comparable to the quality of the unicast path between that pair of members. This implies a requirement for a minimum-diameter tree. But, as the protocol constructs the tree in real time and has no prior knowledge of node arrivals, it is hard to construct this optimum tree. Refinement is a solution to this problem. It moves the overlay structure from the local optimum to the global optimum and improves the system's performance. But excessive refinement makes the structure unstable due to the ad hoc nature of node behavior. Moreover, the effectiveness of the refinement for real-time applications is questionable due to interrupted data distributions among the members. Thus, the designer must carefully choose the depth and frequency of tree refinement for a given ALM.

3.2.4 Routing Mechanisms

Once the overall group management has been designed and the various choices are fixed, the most important part of the design is how the tree (or a different structure) is formed to provide the multicast service. This greatly depends on the previous choices such as application domain (mainly determining the quality metric and constraints), deployment level (mainly determining the resources available to each node in terms of permanency and bandwidth) and group management. Design of the routing mechanism typically involves a (heuristic) solution to a graph theory problem. That is, given a certain graph (i.e. a certain existing structure of nodes) and certain constraints on each node (e.g. inbound and outbound bandwidth constraints), the problem involves the creation of a structure connecting the group of users (or in case of a tree, connecting a source to all its recipients) that satisfies a given requirement-e.g., minimum overlay delay or minimum worst-case delay. The solution to the problem largely com-

prises the routing mechanisms; the routing mechanisms must then be augmented with stipulations about nodes leaving the multicast structure, as well as possibly periodic or event-based refinement strategies for the improvement of the structure. In this section, we include a survey of common approaches to the routing mechanism.

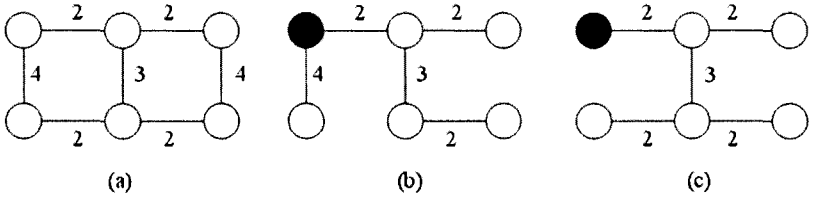

Figure 3.6: (a) A graph with link costs (b) Shortest path tree (b) Minimum spanning tree

Group 1 - Shortest Path

The aim of this group is to construct a degree-constraint minimum-diameter spanning tree. Here, it can use round-trip time (RTT) to determine the shortest path tree from the source to the end-hosts and can minimize delay for applications while considering the degree constraint and QoS. A Shortest Path Tree (SPT) constructs a minimum-cost path from a source node to all its receivers (see chapter 25 of [CLR93] for Dijkstra's algorithm for building SPTs). The shortest path tree or its variants is commonly used in ALM protocols (such as Yoid [Fra99], SpreadIt [HDGM01], TAG [KF02], RITA [XTBL03]) in order to construct a source-specific multicast tree or, in graph theoretic terms, a rooted tree. Figure 3.6b shows the SPT rooted at the filled-in node. It is important to note that both MST and SPT can be modified to respect the degree constraints of each node [BV95].

Group 2 - Minimum Spanning Tree

This group does not worry about degree constraint of nodes and just tries to construct a 'low-cost' tree or, in other words, a Minimum Spanning Tree. Given a graph with a cost associated with each edge (usually delay), a Minimum Spanning Tree (MST) is a tree with minimum total cost spanning all the members (see Chapter 24 of [CLR93] for Kruskal and Prim's algorithms for building MSTs). Given the graph with edge costs shown in Figure 3.6a, an MST is constructed to have the minimum total cost as shown in Figure 3.6c (total cost is 11 in this example). An MST is commonly used by a centralized ALM protocol such as ALMI [PSVW01] and HBM [RES01] in order to construct a low-cost shared tree that is not rooted at any particular source (a shared tree implies that all nodes use the same tree to distribute their data).

Group 3 - Clustering Structure

This group constructs clusters of nodes that can be used to construct trees. In order to better organize the overlay tree and reduce control overhead, some ALM protocols such as ZIGZAG [THD04] and NICE [BBK02] construct a hierarchical cluster of nodes with each cluster having a "head" representing it in the higher layer (Figure 3-7). The advantage of a hierarchical clustering approach to a multicast tree is the reduction in control overhead (nodes keep states only about a subset of other nodes) and faster joining and management of the tree at the cost of a sub-optimal tree and a lack of hard guarantees on the degree limitation of each node.

Group 4 - Peer-to-Peer Structure

In P2P structure, the routing is simply done through reverse-path forwarding or forward-path forwarding, or in some cases a combination of both. We can say that many ALM protocols (such as RMX [CMB00]; Gossamer [Cha00]; Bayeux [ZZKK01]; Borg [ZH03]; Scribe [CDmKR02]) operate based on an existing peer-to-peer substrate

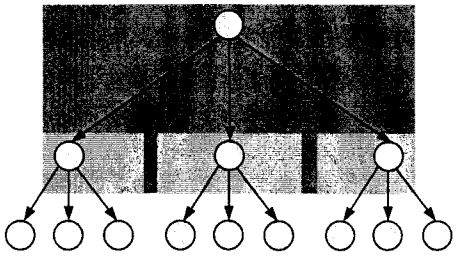

Figure 3.7: A hierarchical cluster of nodes with cluster size 4

that serves as a mesh on top of which an overlay multicast tree can be constructed, using either a reverse-path forwarding scheme (Gossamer [Cha00]; RMX [CMB00]; Scribe [CDmKR02]), a forward-path forwarding scheme (Bayeux [ZZKK01]) or both (Borg [ZH03]). The advantages of these approaches are low control overhead and distributed management of the multicast tree, but they cannot restrict the degree of each node and produce sub-optimal trees.

It should be noted that peer-to-peer technology is a research area of its own. In general, a P2P system is a system where peers communicate directly with one another. As such, there is not necessarily a multicasting component. For example, Skype [2] is a well-known P2P Internet telephony system that does not use multicasting. The P2P aspects covered here applies to ALM systems that have a P2P component, such as TVUPlayer [3] , Sopcast [4] , and PPLive [5] , to name a few.

3.2.5 Degree-Constraint Routing

The feasibility of supporting multicast data over an ALM depends on whether or not there is available bandwidth (out degree) at the end-hosts. Usually, end-hosts

[2]http://en.wikipedia.org/wiki/Skype
[3]http://en.wikipedia.org/wiki/TVUPlayer
[4]http://www.sopcast.org/
[5]http://en.wikipedia.org/wiki/PPLive

have asymmetric downloading and uploading capabilities. Moreover, the heterogeneity of outgoing bandwidth of end-hosts forces protocols to consider realistic degree assignment. It may happen that a user has zero out degree, i.e. that this user is a pure receiver. In the real world, around 50% of hosts have zero out degree to support a streaming bit rate [SGMZ04]. From a practical perspective, asymmetric bandwidth cannot be ignored and should be taken into consideration when assigning out degrees to nodes during implementation. This reflects the maximum bandwidth a node can provide. For example, if a node has an out degree of four, this means it can support at most four children. There are two types of degree constraints. In some cases, there is only a bound for the maximum number of edges that a node can have; this is usually flexible and can be changed according to different applications. In other cases, there is a fixed bound that is restricted and predetermined. Minimum Spanning Tree (MST) and Shortest Path Tree (SPT) routing algorithms can be modified to respect the degree constraints of each node. The problem of finding minimum-cost degree-constrained multicast trees or degree-constrained Steiner trees is NP-complete [Dou92]. There exist several heuristic approximation algorithms addressing this problem [CLR93][KR00][KR03][MLRS02][RMR+01]. Some of these algorithms (such as [KR00][KR03]) do not provide exact guarantees on the degree of each node in the tree and instead provide a bound on the worst-case degree. Others focus on constructing a single tree and do not consider multiple trees over the same graph ([CLR93][ST02][RMR+01]). Though there has been some research into constructing multiple trees on a shared graph [CGY00], they still only provide a bound on the worst case (maximum) degree of any node as opposed to guarantees on the individual maximum degree for every node as is required for a protocol supporting multi-source collaboration applications.

3.3 Challenges

An open issue for all ALM protocols is that of tree refinement: the reorganization or shuffling of the nodes in the tree. This is usually conducted to enhance the system performance. In ALM, the quality of the path between any pair of members is comparable to the quality of the unicast path between that pair of members. Typically, a lower-diameter tree performs better than a higher-diameter tree. Hence, refinement is a way to improve the quality of an ALM structure once it is already constructed. The key point is that, if a node with zero out-degree joins a multicast session, the tree cannot be extended beyond that point which ultimately increases the height of the tree. To handle such situations, refinement acts as a solution. But it is an expensive operation and thus should be applied only under special conditions. This is because protocols require too much information to carry out the operation. Research should therefore be conducted to find efficient mechanisms to determine whether or not refinement is applicable to a particular node. If so, how much it improves the performance of the system-say, in terms of average latency or other parameters. The protocol should also be aware of the transient period of the refinement when it actually takes place-whether it affects its dependent nodes, and if so, to what degree it affects them. Furthermore, the protocol must consider its side effects, including churn, which may lead to an inconsistent system. As an example, OMNI uses local transformations (child promote, parent-child swap, iso-level-2 transfer, aniso+level-1-2 swap) and probabilistic transformations (simulated annealing) to refine its structure [BKK+06]. As it is an expensive operation and requires extra care, frequent refinement may adversely affect the system performance. Most ALM protocols apply refinement operations strategically and rarely.

Another open issue is balancing the two conflicting design goals mentioned earlier: (a) minimizing the length of the paths (usually in terms of hops) to the individual destinations and (b) minimizing the total number of hops required to forward a packet

to all its destinations. The minimum spanning tree (MST) and the shortest path tree (SPT) are two well-known data distribution methods in ALM. The MST optimizes the resource usage of the multicast tree but the pair-wise paths may not be optimal and can cause large end-to-end delay. Hence, it is suitable for non-interactive data dissemination when end-to-end delays are not an issue. In SPT, the distribution tree will consist of separate unicast connections from the sender to each receiver. It is optimal from the source to the receiver in terms of end-to-end delay, but it causes high consumption of network resources. Moreover, it is not practical when the sender's bandwidth is not sufficient to serve all receivers simultaneously. Scalable ALM systems usually require clustering of the nodes. This hierarchical clustering has low control overhead as nodes keep states only about a subset of other nodes. Furthermore, faster joining and group management is possible at the cost of a sub-optimal tree.

In this chapter, we looked at the roots and rationale of application layer multicasting. Compared to IP multicasting, ALM has certain disadvantages such as longer delays. However, due to its overwhelming advantages for certain applications, such as immediate deployability and application-specific adaptation, it can be a practical solution to many of the existing problems in multi-user communications. The fact that an ALM protocol can be developed and deployed on the Internet without the need to make any changes to the existing network infrastructure and the ability to evolve and apply modifications to the protocol quickly and easily at the application layer have helped give the ALM approach a quicker start compared to other multi-user communications solutions. These advantages have caused the serious consideration and development of ALM protocols, which in turn would lead to the creation of new applications and communications paradigms on the Internet.

The popularity of application layer multicasting continues to grow in different fields as an alternative to native IP Multicasting. These include newsgroups, videoconferencing, internet games, internet jukeboxes, interactive chat-lines, distance

learning, and video on demand, to name a few. Although ALM has been considered an active research topic over the last decade, still there are many open issues for research into creating efficient and robust ALM protocols in terms of application requirements and better quality of service.

Chapter 4

The Hybrid MMOG Architecture

In this chapter, a new hybrid (P2P combined with Client-Server) architecture designed for zonal MMOGs will be presented. As will be shown, this solution will address some of the problems inherent in the related work presented in chapter 2, such as scalability and discontinued zone views. However, since our work is partially client-server based, the single point of failure issue remains unresolved. While existing fault tolerance approaches such as back-up servers can be used to overcome this problem, their discussion and details are beyond the scope of this thesis.

We will also present a number of load balancing schemes for zonal MMOGs in the next chapter, among which Two-Level Partitioning (section 5.1) and Multilevel Multiphase Load Balancing (section 5.2) can be applied to the architecture proposed here, with the latter being more efficient as will be shown. Let us start by an overview of the proposed architecture and assumptions that are made.

4.1 Architectural Overview and Assumptions

To accommodate a large number of players and to ensure proper game administration (e.g. intra-zone and inter-zone communications, load management) the proposed

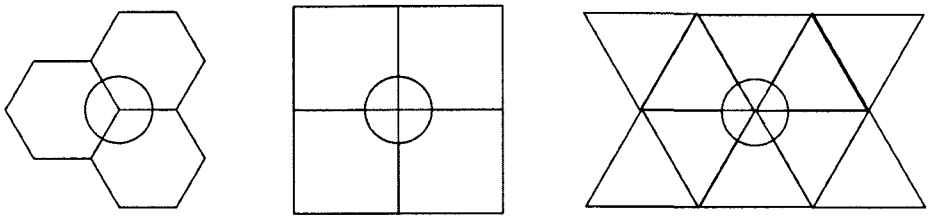

Figure 4.1: Number of zones covered by a visibility circle in hexagonal, square, and triangular zones (respectively from left to right).

architecture divides the virtual world into several manageable logical zones where each zone covers the players in a given vicinity. In this thesis, the terms "zone", "region", and sub-space are interchangeable. Each zone will have one "master", essentially a server, responsible for administrative tasks that will be described shortly. Zones in our architecture are hexagonal. Other options are square tiling or triangular tiling. We choose the hexagonal tiling because of its geometric properties that have also made it appropriate for use in wireless environments, cell phone cites, or any other partitioned environment that must deal with mobility of entities. What is unique about hexagons is that, when considering a player with a certain radius of visibility, the maximum number of different zones covered at a given time is 3, as opposed to 4 for squares and 6 for triangles, as shown in Figure 4.1. This means less overhead in communications between zone masters when a player is near multiple zones and hence needs to see what's happening inside its neighboring zones, as will be discussed in 4.6

For the game world itself, we assume a virtual 'planar' world in which players move over the surface but are not restricted to stick to the surface. The geometry of the world does not allow teleportation; i.e., there is no special gate that connects different isolated parts of the world. In addition, this framework makes the following assumptions:

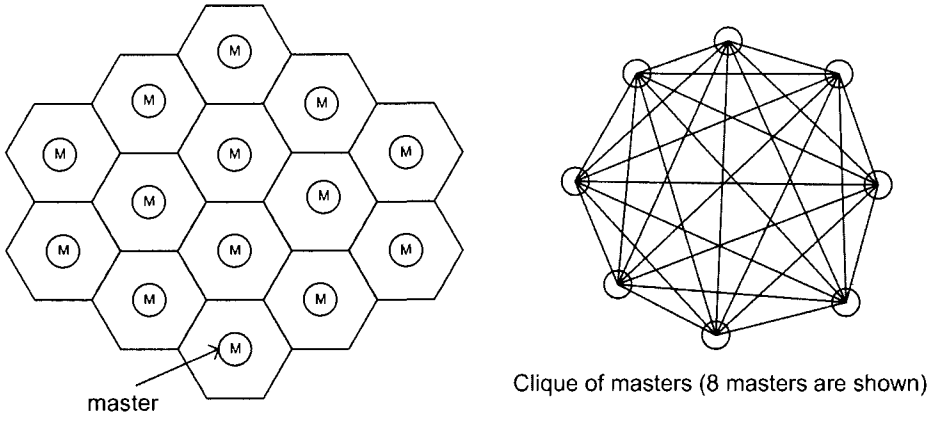

Figure 4.2: The top level hybrid architecture

- each player (here the terms player, peer, entity, user, object, avatar are used interchangeably) has a unique identity, such as its IP address;
- each player is categorized into a class, also called an object type, based on certain characteristics such as velocity (more about this in section 4.2);
- each player has an interest vector to define its inclination (or interest) in different other object classes;
- each player has a maximum outgoing bandwidth for a game session, used to determine its out degree (number of peers it can support as a parent), which is also known as the fan out;
- each player knows the identity of at least one master at the startup;
- Masters know of each other (configured by a human operator) and can exchange messages with one-another.

The general hexagonal zone layout with a master in each zone is shown in Figure 4.2. The masters comprise the top-level hierarchy, are connected to each other, and

can exchange information when needed. A master performs several administrative tasks coordinating players in the virtual world, and provides a consistent game world. The key functionalities of the masters are outlined below:

- Player registration, which takes note of which players are currently in this zone;
- Zone organization; i.e. defining zone size and buffer region at the beginning of the game and sharing of game states among the masters of the neighboring zones;
- Construction of an Application Layer Multicast (ALM) tree among players of the same group in a given zone;
- Inter-group communication, i.e., message passing between P2P groups within a given zone, which is needed when players in different groups are in close proximity to each other;
- Inter-zone communications; i.e., enabling a player to communicate beyond its own zone, which is needed when players in different zones are in close proximity of each other;
- Player hand-off from one zone to another when the player crosses zone boundaries; and
- Load balancing, as will be shown in chapter 5.

A set of master nodes regulates the operation of the MMOG, while each individual master also provides overlay services with the active participation of the players in its zone. In that sense, the system is hybrid as it combines the benefits of both centralized and distributed systems. To overcome the functionality limitations of the IP multicast, application layer multicasting (ALM) has been chosen for intra-zonal communication. The advantages of such an approach were presented and explored

extensively in Chapter 3. In this model, most of the visibility issues and game functionalities are solved by the local ALM structure and through the master, who is also a member of the ALM tree. In addition, the master can learn the state of other zones through the exchange of explicit messages with other masters when needed.

After this overview, we now proceed to the detailed description of the system. The rest of this chapter will cover:

4.2 Clustering of Players: how, within the same zone, players of different type are grouped together in order to stabilize the ALM trees inside that zone;

4.3 Intra-zone Communications: how the ALM tree is formed between players of the same group, and how communications is performed;

4.4 Message Overhead Reduction: how to reduce bandwidth and processing by eliminating message delivery between players who, even though they are in the same group, they are not within each other's visibility radius;

4.5 Zone Crossing: how to stabilize connection/disconnection rate between a player and multiple masters when the player crosses a given zone boundary back and forth repeatedly, which can happen during a battle in repeated shoot-and-run, return and shoot-and-run, ... fights;

4.6 Visibility Issues: how to allow a player to "see" inside another zone when the player's visibility crosses into that zone; and

4.7 Seamless Player Handoff: how two masters should hand off a player from one master to the other without the player noticing any discontinuity or losing any updates.

4.2 Clustering of Players

To the best of our knowledge, current MMOG systems do not differentiate between soldiers walking slowly and tanks driving fast when they construct their routing table. This assumption of likeness is inappropriate and can lead to unstable overlays. For

example, assume the high-velocity players are positioned at the top of the ALM tree structure. These fast-moving players can leave the zone soon, breaking the links with their children and causing reorganization of the players in the ALM structure. But if there are clusters in each zone, a leaving player only affects the dependent nodes of that cluster. This simply isolates different types of players in the structure and shifts the zone-crossing penalties from the whole zone to a cluster. This is the reason that we distribute players into multiple clusters based on their physical characteristics like velocity or movement pattern. As the velocity of an avatar is not constant, we consider average velocity to determine that player's class type. It follows that clustering of players on their activities or types can significantly improve the performance of zonal MMOGs [ASO07]. The reason is that clustering forms a number of P2P overlays and restricts, for example, slowly moving players to be children of fast moving payers and therefore from being affected by the departure of the latter from the P2P data delivery path. This means a leaving player can only break routing paths within its own cluster keeping other clusters untouched. In other words, the P2P routing problem faced due to a player disappearing is limited to a cluster. Clustering will therefore help stabilize the overlay networks used in zonal MMOGs.

Our model assumes the general characteristics or attributes of the players are invariant; and based on this assumption it classifies players and forms clusters. But during runtime, a player's attribute might change, either temporarily or permanently. For example, a slow-moving soldier can jump into a jeep and drive away at a fast speed. In this case, the master can move such players from one cluster to another and refine the overlay network accordingly for better stability of the system. Details for this are however beyond the scope of the presented model.

Even though players are clustered, state information about a player must be relayed to other clusters. For example, state messages from cluster C4 in Zone 1 must be sent to clusters C1, C2, and C3 in the same Zone 1 (Figure 4.3) but not necessarily to the other zones like Zone 2. We used the term "'not necessarily"

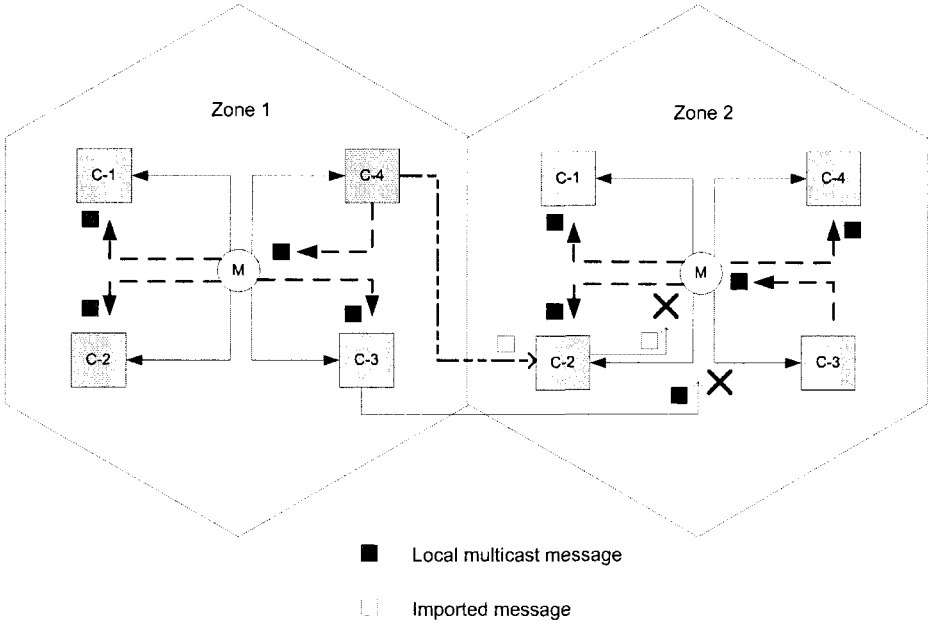

Figure 4.3: The clustered ALM, message propagation and isolation scenery

because there can be cases when this is necessary. For example, players staying in different zones can be in each others' close proximity and "visually" can see each other. This requires solving visibility problems to give a continuous view. But such "imported" messages from a foreign zone must be filtered considering their irrelevance to local zone members. For example, a player from cluster C2 in Zone 2 imports a message from a player from cluster C4 in Zone 1 with the help of the master (Figure 4.3), because the two players are very close to each other. This imported message is not flooded further in Zone 2.

4.3 Intra-Zone Communication

Intra-zone communication means communication within a zone or inside a group. The players who are closer to each other in the virtual world interact more frequently and

are involved in many common activities. In order to retain a consistent game space, there is a need to exchange messages among them, and intra-zone communication becomes necessary. For local communication inside a zone, zone master's and players' active participations are integrated to overcome the resource limitations of the master. This is a kind of overlay-based state-sharing mechanism. In our system, a graph theoretic framework is considered to create an ALM tree for players. We incorporate the features of *dominating set* (explained in 4.3.3), which is a well-known routing approach in wireless ad hoc networks. The reason for using dominating set is its cooperative communication structure in ad hoc environments and distributed fault tolerance property, which fits quite well with an MMOG scenario. As it cannot be directly applied to our system, we adapt it as will be shown in section 4.3.3. There are two steps to forming the ALM tree: *constructing a mesh* and *defining a routing table.* A greedy heuristic algorithm is run to build a mesh with the objective of minimum cost-connected graph subject to degree constraints. This ensures players are not out of resources while the game is on. The goal is to discover routing paths over the mesh to reduce high network latency and to reduce redundant network resource usage over other existing scalable approaches.

4.3.1 General Policy and Node Registering

In this framework, each player requires registration through a master. The player provides the following information to the master:

- class type [i.e. slow, normal, fast or very fast]
- interest vector to represent its inclination (or interest) to other types of players
- its available outgoing bandwidth to determine its out-degree

The 'type' is used to differentiate players in a zone. The master accepts a player's join request and places the player in the appropriate cluster. As described in section

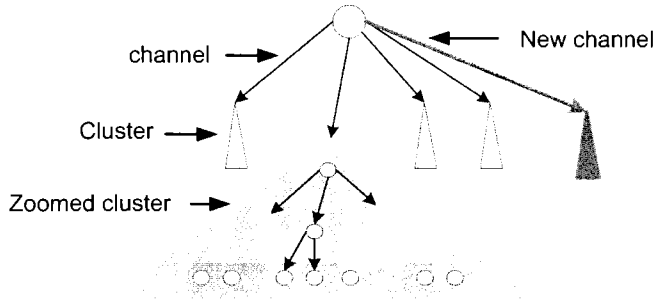

Figure 4.4: Controlling the diameter of the tree from master's perspective

4.2, the clustering technique never allows a slowly moving player to suffer from the activities of other outgoing normal, fast or very fast-moving players as there is no differentiation in the ALM path path between any two players of same type. We use average velocity to classify players into types.

In our architecture, each cluster will be served by a number of channels, where each channel has its own separate ALM tree. The reason not to use one ALM tree (and hence only one channel) for the whole cluster is that, if there are too many players in one cluster, putting all of them on one ALM tree will make the tree's diameter large and might exceed the required end-to-end delay thresholds. By separating a cluster into a number of channels, each channel's ALM tree will meet delay thresholds and give a higher quality of service. It is technically important to define how many channels can be dedicated to a cluster type. MM-VISA does not fix the number of channels early - it determines this as players join as will be discussed next. Let us assume, we have T types of objects/classes; thus the master should have at least $T \times b$ bandwidth to make the system operational where b (application-specific parameter) stands for the bandwidth required by the master to serve a single channel. Thus, if we define the master's channel capacity (C) as the number of clusters it can support simultaneously, then, we must have $C \geq T$. Let $outGoingBW$ be the master's total available bandwidth. So, $C = \lfloor outGoingBW/b \rfloor$. For a specific game, b is fixed

(depends on the game), $OutGoingBW$ depends on the server/network resources used, and T can be configured. Hence, C is known when the game starts. When a new join request comes to the master, the master processes the request and accommodates the node by adding it to the ALM tree of the appropriate cluster. To reduce average end-to-end delay, we restrict the height of the ALM tree. Whenever a channel can no longer support more players (ALM path exceeds its maximum limit), the master opens another channel as long as it has enough resources. Figure 4.4 presents such a tree, where players under a particular channel have the same attributes.

4.3.2 Mesh Construction

The master of each zone constructs a mesh, used later to create the ALM tree, based on geographical position. Instead of using end-to-end delay or hop count, geographical position is used as an alternative to perceiving physical distance on the fly. The key benefit of geographic location comes from its quicker approximation as we do not need to exchange explicit overhead messages among players to calculate distance or delay. The master determines the player's fan-out, i.e. out-degree, using the following simple division where b_c stands for the bandwidth needed to serve each client: $out-degree = \lfloor uploadBW/b_c \rfloor$, where $uploadBW$ is the client's upload bandwidth. The master is a member of all meshes and therefore all meshes are connected. The mesh construction procedure is explained below.

If we allow each node to choose the closest nodes as its neighbors, the resulting graphs may be disconnected. Consider Figure 4.5, where Nodes a, b, c and d each have a degree of two. Nodes a, b and c are close to each other and use their degrees. But Node d cannot make an edge to any of them, as none of a, b, or c have any capacity to serve d, so the resultant graph is disconnected. This could have been avoided if the algorithm was not random, and instead of c connecting to a it would first connect c to d and then d would connect to a. To remove the randomness and

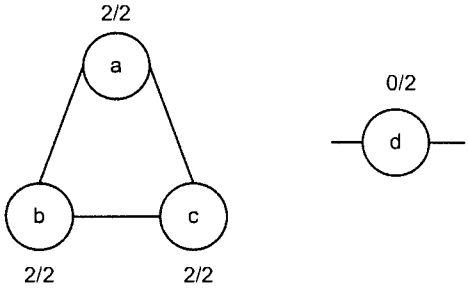

Figure 4.5: Random choice leads to disconnected graph

avoid a disconnected graph, we propose a degree-constraint minimum-cost connected mesh construction algorithm. The objective of this algorithm is to determine the edges to form a connected mesh subject to the degree constraints while optimizing (minimizing) the overall distance. This algorithm works in 2 steps. In the first step, considering a completely connected graph, the master sorts the edges in the ascending order.

It greedily selects $N - 1$ edges (N being the number of nodes) to span all the members. This policy ensures a connected graph - in fact a tree - while satisfying the degree constraints. The next step is to include as many edges as possible to form a dense mesh while obeying degree constraints and greedily optimizing overall distance. Let us take a look at the details of this algorithm next.

4.3.3 Data Delivery Path Based on Dominating Set

A routing table is a set of rules or database usually kept in a table format that contains the necessary information to forward a packet along the path towards its destination. This section describes how to construct such routing tables to share game states among the players in the overlay network. In our model, these routing rules will tell which player will forward packets to whom according to the dominating

Algorithm 1 Constructing meshes for a zone: a heuristic approach

Require: $G = (V, E)$ for each cluster

Ensure: *meshes*

1: **for** each cluster **do**
2: $e :=$ minHeap(E) {Let $e := edge(n_i, n_j)$}
3: add edge e to G'
4: $vertexSet := \{n_i, n_j\}$
5: // N : total no of nodes in a cluster
6: **while** |vertexSet| $\neq N$ **do**
7: $e :=$ minHeap(E)
8: **if** $(n_i \in vertexSet \oplus n_j \in vertexSet)$ **then**
9: **if** ($n_j \notin vertexSet$ and $n_i.freeDegree > 0$) **then**
10: $vertexSet := vertexSet \cup \{n_j\}$
11: add edge e to G'
12: **else if** ($n_i \notin vertexSet$ and $n_j.freeDegree > 0$) **then**
13: $vertexSet := vertexSet \cup \{n_i\}$
14: add edge e to G'
15: **end if**
16: **end if**
17: update data structures
18: **end while**
19: greedily add as many edges as possible to G subject to degree constraints
20: **end for**

set principles. In graph theory, a dominating set for a graph $G = (V, E)$ is a subset V' of V such that every vertex not in V' is joined to at least one member of V' by some edge. Minimum dominating set refers to a dominating set where the solution is optimal, meaning the fewest number of nodes have formed a backbone covering the entire graph. However, to find a minimum dominating set is indeed NP-hard [1] [WL99], but a minimal set can be found by using efficient approximation algorithms. In routing, it is effective for a dominating set to be minimal, as a minimal number of nodes cover every nodes of the graph and the backbone will be less complex to build and maintain.

A graph is a set of nodes or vertices connected by links called lines or edges. In an undirected graph, a line from point A to point B is considered to be the same as a line from point B to point A. For our overlay network, we construct an undirected graph $G = (V, E)$ using the heuristic approach shown in algorithm 1. The objective is to determine a subset of nodes that would connect all other nodes in the overlay. These nodes will be the overlay "routers" which we will call *gateway nodes*, and which will ultimately form a backbone disseminating packets across the overlay. This subset can be determined using the dominating set rules which will be explained in this subsection.

Let us define two terms. In a graph, the open neighbor set of a vertex v, represents a set of all the direct neighbors of v whereas the close neighbor set is the union of the open neighbor set and the vertex v. Mathematically, open neighbor set and close neighbor set of a vertex $v \in V$ are represented by $N(v) = \{u|(v, u) \in E\}$ and $N[v] = N(v) \cup \{v\}$ respectively, where vertex u and vertex v are two ends of an edge $(v, u) \in E$. The following marking schemes are carried out to determine the backbone nodes that connect the other nodes together (i.e. gateway nodes). Nodes are labeled either as T or F, standing for gateway and non-gateway nodes,

[1] The class of decision problems that are fundamentally harder than those that can be solved by a nondeterministic Turing machine in polynomial time.

respectively. A function called marker $m(v)$ assigns T or F to node v depending on whether v is gateway or not. Initially, the algorithm assigns F to each $v \in V$. While the dominating set algorithm runs the marker $m(v)$ is changed to T, if there are two unconnected neighbors of v.

A subgraph G' of a graph G is said to be induced when, for any pair of vertices u and v of G', uv is an edge of G' if and only if uv is an edge of G.The subgraph G' is induced by V' where $V' = \{v|v \in V, m(v) = T\}$ and, according to the definition of dominating set given earlier, the vertex set V' becomes the dominating set. However, the dominating set constructed in this way is not minimal. Subset relation techniques like set covering can be used to reduce its size to make it more minimal. Hence, the next two rules are applied assuming that each vertex v has a unique identifier named $key(v)$, which for example can be a hash of its IP address:

RULE 1: *Consider two vertices u and v in G'. If $N[v] \subseteq N[u]$ in G and $key(v) < key(u)$, mark it as non-gateway node if node v is a gateway node, i.e. G' becomes $G' - \{v\}$.*

RULE 2: *Assume u and w are two marked neighbors of a marked vertex v of G'. If $N(v) \subseteq N(u) \cup N(w)$ in G and $key(v) = \min\{key(u), key(v), key(w)\}$, mark it as non-gateway node if node v is a gateway node, i.e. G' becomes $G' - \{v\}$.*

It can be shown that: (a) If G is connected but not completely connected, then V' forms a dominating set of G; (b) the induced graph G' is connected; and (c) the shortest path between any two nodes does not include any non-gateway node [WL99]. Figure 4.6 shows a connected dominating set construction process. Once the connected dominating set is constructed, it defines the core nodes responsible for data forwarding. The master sends the list $N[v]$ and their labels (gateway/non-gateway) to each node $v \in V$.

We use a mesh-first approach to construct the routing substrate, because this is more fault tolerable than a tree-first approach, as discussed in chapter 3. In the mesh-first approach, the members maintain a connected mesh topology. Typically,

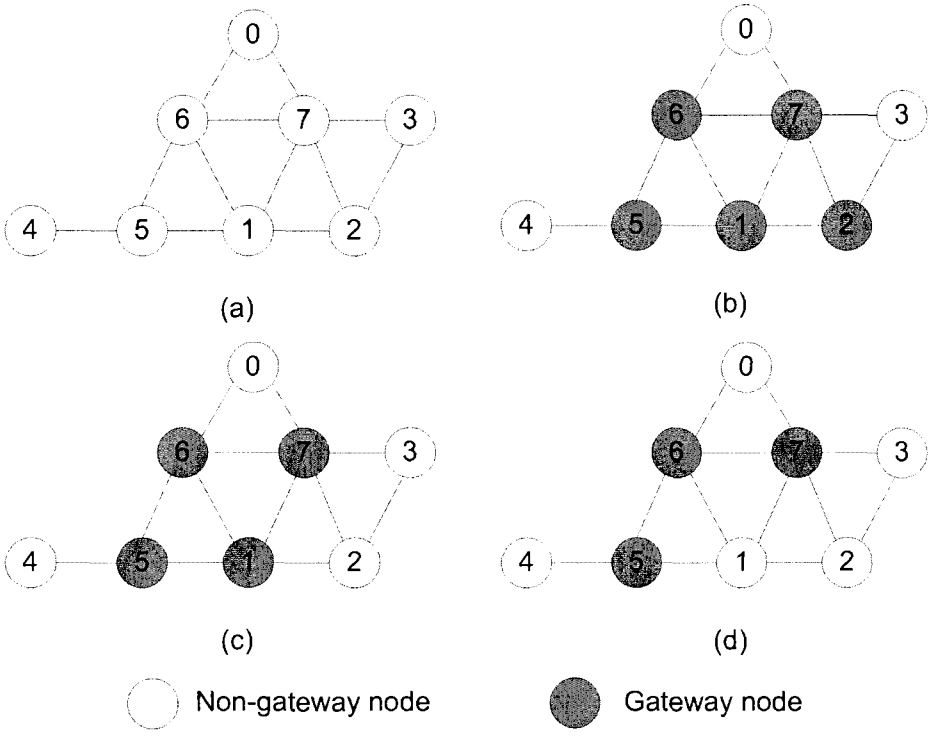

Figure 4.6: Connected dominating set formation: (a) Initially all nodes are non-gateway nodes, (b) A node becomes gateway if it has two unconnected neighbors, (c) Rule-1, (d) Rule-2

the source is chosen as the root and a routing algorithm is run over the mesh relative to the root to build the tree. This mesh topology is explicitly created at the beginning; hence, it is known. On the other hand, as explained in chapter 3 for mesh-based approaches, the resultant tree structure is unknown at this point, but the quality of the tree depends on the quality of the mesh chosen.

After having the mesh, the next task is to determine data delivery paths for the nodes. To make the system practical, each node exchanges the neighbor set only with its own neighbors. This requires a route activation message `RouteActMsg`) for loop-free routing. Each source sends a periodic `RouteActMsg`. When a node receives

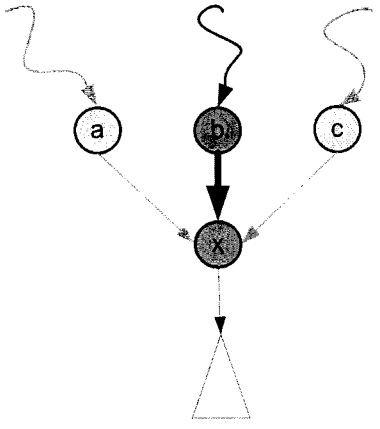

Figure 4.7: The ignore set buildup

more than one activation message through different paths sent by the same source, it keeps one and ignores the others, denoted as ignore set $I(v)$. In Figure 4-6, Node x receives three packets from the same source node. According to Figure 4.7, Node x accepts only one and declines other two. Here, the ignore set is $I(x) = \{a, c\}$. The routing policy is simple: if a non-gateway node receives a packet, it does not forward it to others, but if a gateway Node u receives a packet from Node v, it forwards it to $w \in F$ where $F = N(u) - N(v) - I(u)$.

4.3.4 Handling the Ad hoc Nature of Nodes

Incorporating newcomers and handling early departures are also complex tasks in this type of application. In the following subsections, we explain a simple but effective algorithm to treat such situations.

Newcomers

Upon receiving a newcomer's `Join Request`, the master discovers a neighbor set based on the requester's location and available bandwidth, lets it join the session,

and informs its neighbors about the new node. When the system is operational, changing the overlay structure could be costly because topology reformation will lead to break in synchronous communication between players. But because we use the dominating set, each node can independently determine its type (gateway or non-gateway) simply by exploiting its neighbor set with the help of a unique identification number (i.e. key). The tree is then reformed according to the algorithm presented in 4.3.3. We will see more details about this in the recovery action: section below.

Early Departures

Ideally, departures can be divided into two categories: *graceful* and *ungraceful.* In a graceful departure situation, a departing node notifies its communication partners and then leaves. The nodes can update their states and can reconstruct the structure. In an ungraceful situation, a node leaves without a notice (e.g., a computer crashes). To handle such cases, every node runs a node departure detection mechanism. This mechanism is similar to the fault-detection algorithms, where a "keep alive" message is sent to a communication partner and a timer is started. If the timer elapses and it does not receive any acknowledgment, the node realizes that the partner has departed and then acts accordingly.

Mesh Reconstruction

The recovery process follows two steps. The first step fixes the neighbor set and the second step deals with updating the routing responsibilities, i.e. flipping from gateway to non-gateway or vice versa if necessary. After the detection of a node departure, let Node a, the "highest key node" neighbor of Node a (Node $HKey$) informs the incident to the master, where Node $HKey = max\{key(u)|u \in N(a)\}$. This means that only the highest-key neighbor is responsible for sending such messages to the master. This is possible as every neighbor $u \in N(a)$ knows the neighbor set $N(a)$.

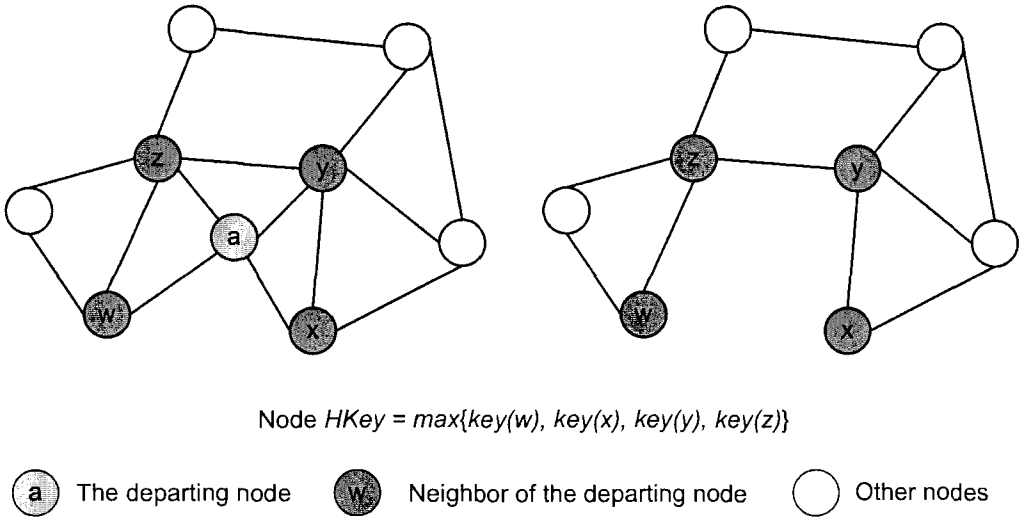

Figure 4.8: Handling node departures

This policy discards the redundant repair requests. The master starts repairing the mesh by simply dropping the links (a, u) where $u \in N(a)$ and greedily including edges incident to nodes $u \in N(a)$ (Figure 4.8). Let R be the set of nodes affected due to the mesh reconstruction. The master sends the updated information to the nodes $r \in R$ to update their neighbor lists.

Recovery Action

The recovery action is carried out in a distributed manner. Every node $u \in R$ (R being the set of nodes affected due to the mesh reconstruction) follows the following steps, in order, after receiving the modified neighbor list from the master for a consistent system: (1) u marks itself as a non-gateway node; (2) u marks itself as a gateway node if there are two unconnected neighbors; (3) if u is a gateway node, apply rule 1 to become a non-gateway node; (4) if u is still a gateway node, apply rule 2 to become a non-gateway node. After recovery, nodes forward packets based on the algorithm presented in the previous section.

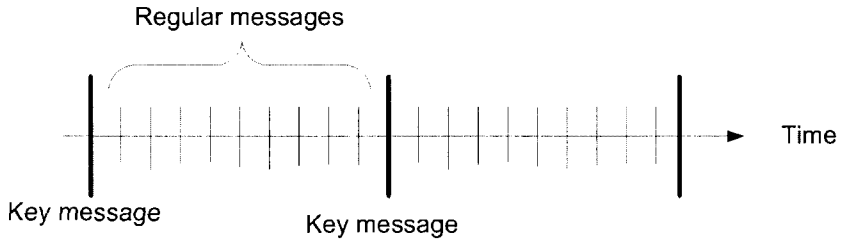

Figure 4.9: Key and regular messages in the timeline

4.4 Message Overhead Reduction

Based on the position of players within a zone, two nodes belonging to the same cluster may actually not be in each other's visibility range; thus, not all messages are important to all the players. In the following paragraphs, we discuss a simple but effective mechanism to reduce such undesirable message propagation.

As the simulation progresses, events are shared among the players in a zone using the P2P structure. These messages are short but frequent, so filtering out messages that are outside a player's visibility range will further reduce bandwidth consumption on the ALM tree. We consider two types of messages: *key messages* and *regular messages*. Key messages are important ways to refresh and synchronize game states within a zone. They are typically associated with terminal events, such as a sniper firing a bullet (another update representing this event will not be sent) or a vehicle stopping at a point (no update messages will be sent until it moves again). These are 'key' because losing those messages can cause inconsistency. On the other hand, messages between two key messages are defined as regular messages; they are not terminal messages and the loss of one is made up relatively quickly by the arrival of the next one. Figure 4.9 shows the position of key and regular messages.

To accommodate such filtering, we define two rules. First, a gateway node is never allowed to filter out messages, because gateway nodes work as a backbone that

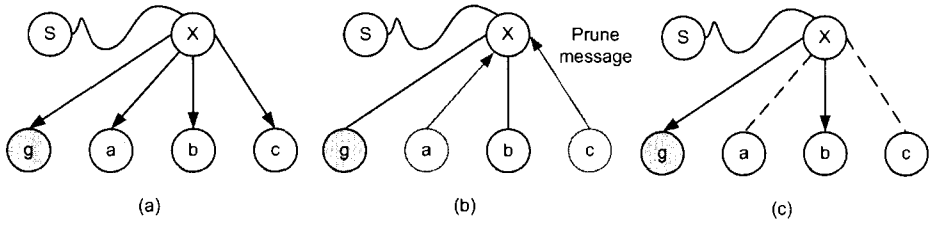

Figure 4.10: (a) Propagation of key message (b) Prune activation message (c) Filtering

delivers messages across the overlay. Second, a non-gateway node can request, from its parent, pruning of regular messages from a given source, but key messages will still be delivered to it in order to keep it up-to-date regularly in case a previously-pruned source does move into the node's visibility range. Let us illustrate this by an example. Consider Figure 4.10a, where Source S is sending messages in the overlay tree. Let us assume a key message from S reaches gateway Node x. Although Nodes a, b and c are part of the same ALM tree as S, they may not be interested in S if their visibility range does not encompass S. Let us assume this is the case for a and c. As Node x gets a key message, it forwards it to all nodes according to the routing policy mentioned earlier. Now, Nodes a and c send prune messages to Node x, requesting pruning on source S (Figure 4.10b). Thus any regular message originating from Node S will not be relayed to Node a and Node c (Figure 4.10c). However, the ensuing key messages from S will still be sent to a and c in order to update their area of interest status. Table 4.1 presents a modified routing table for Gateway node x with respect to Source S. This scheme greatly reduces unnecessary message propagation in the overlay structure as will be shown in Chapter 7.

Table 4.1: Routing table for Gateway Node x

Source S, Message Type: `Key`

Neighbor	***Node type***	***Pruned***	***Relay***
a	Non-gateway	Not applicable	Yes
b	Non-gateway	Not applicable	Yes
c	Non-gateway	Not applicable	Yes
g	Gateway	Not applicable	Yes

Source S, Message Type: `Regular`

Neighbor	***Node type***	***Pruned***	***Relay***
a	Non-gateway	Yes	No
b	Non-gateway	No	Yes
c	Non-gateway	Yes	No
g	Gateway	Not applicable	Yes

4.5 Interest-driven Zone-crossing

A zone crossing occurs when an avatar crosses a zone boundary, i.e. a node leaves a zone and enters into a neighboring zone. This has an impact on the P2P structure as nodes in the overlay tree are displaced. How well the protocol handles zone crossings will have a direct effect on synchronous communication and hence the quality of the game. There are two tasks associated with zone crossing: first, the detection of zone crossing; and second, the reconstruction of the P2P tree in both the departing and the entering zone. Irrespective of an overlay structure, when a node crosses a zone, all dependent nodes lose the continuity of data as shown in Figure 4.11. This can

cause low quality of experience for users. What is even worse is that a player might do this repeatedly; i.e., a player might move into one zone (Z_{new}) only to come back to its old zone (Z_{old}) within a few seconds, and then move to Z_{new} again, and re-enter Z_{old} again, etc. We can call this the zig-zag effect, which is not unusual to happen in hit-and-hide battle scenarios.

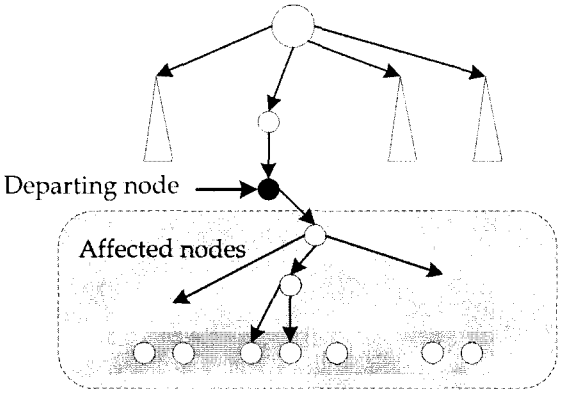

Figure 4.11: Impact of zone crossing

As it is difficult to predict players' movements at the boundaries, repeated connections and disconnections may be encountered either among the zone masters (i.e. servers) or among the multiple overlay networks. VELVET's area of interest management scheme can be implemented to avoid the problem of a player's frequent movement around the zone boundaries [OG03]. Interest-driven zone crossing with dynamic shared regions between adjacent zones is a nice solution to regulate such ungraceful events. Here, each zone has two marks, namely check-in and check-out (Figure 4.12a). The area between the two marks is called the buffer region (a.k.a. common area or overlapped area). It can control the total number of disconnections and connections between the master and a player by adjusting inner and outer marks (Figure 4.12a). To make it even more effective, we propose to integrate an "interest vector" with dynamic shared regions, as described below, taking also into considera-

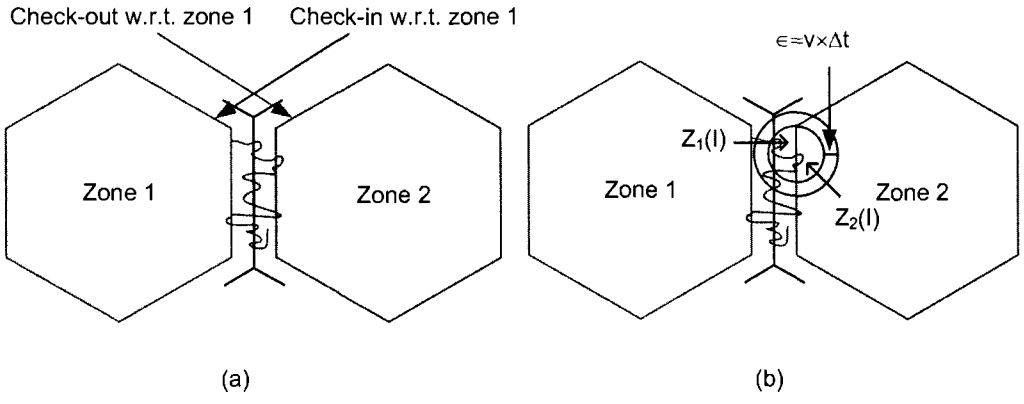

Figure 4.12: (a) Hexagonal regions with check-in and check-out radii with dynamic adjustment of zone marks (b) Controlling of frequent zone crossings

tion the player's velocity; i.e., the overlapped region will be different for different types of players. The interest vector I is defined in the weighted form $I :< w_1, w_2 \ldots w_c >$ where w_i represents the weight of the object of type i, and c is the number of object types with the restriction that $\sum_{i=1}^{c} w_i = 1$. The weights are set by the players when they join and it is up to the application to determine their exact values. Generally, a player can put a bigger weight on its own class compared to other classes. The logic is as follows: first, if a player is completely inside a zone it is a member of that zone, which is obvious. But if it overlaps two zones and crosses out the check-out mark, then the master applies the interest vector formula below to determine the interest values for both zones:

$$Z_j(I) = \sum_{i=1}^{c} w_i \times O_i^j \tag{4.5.1}$$

where $Z_j(I)$ indicates the interest of this node in zone j, and O_i^j is the number of objects of the type i in Zone j. These values will depend on the number of players that fall inside the visibility range of the node and weights of its interest vector (Figure 4.12b). So if $Z_1(I) > Z_2(I)$, the player is considered to be a member of

Zone 1, even if it 'physically' lies in Zone 2. For more overlapping zones (at most three in the hexagonal architecture), the same principle applies. As it is difficult to predict the movement of a player, a safety margin can be considered that expands the area of interest of the concerned player when the master takes the zone crossing decision. This increases the radius of area of interest by epsilon (ϵ), which we define as $\epsilon = v \times \Delta t$, where v is the velocity of the player, and (Δt) is the safety period: a time limit big enough to make decision about zone crossing. The value of Δt is set by the master and is configurable at the beginning of the game. Thus, by controlling the parameters, the protocol can change the circle (as shown in Figure 4.12b by the blue and brown circles) and hence regulates zone crossings.

4.6 Visibility Issues

Passing messages between zones is generally called inter-zone communication. To provide continuous view for players who are either crossing from one zone to another or are close to zone boundaries and hence should be able to see players in neighboring zones, inter-zone communication is mandatory. Consider a hexagonal zone layout where a player stays close to the corner of a hexagon; logically it should be connected to the three different overlays because it can see players close to it in all three zones. Bu connecting to three zones at once leads to overhead, as each message must be shared with three overlays, but most of the other players are likely not interested in those messages. To avoid such irrelevant state-sharing, we follow a hierarchal architecture. The lower layer of this hierarchy is the overlay consisting of the players coordinated by a master solely within a zone, while the upper layer is the mesh formed among the masters. Such a mechanism was roughly described earlier in Figure 4.2.

If a player's visibility exceeds the border of its current zone, the simplest solution is to send an explicit message to the local master, as shown in Figure 4.13a. This local master communicates with the foreign master to solve the visibility problem by

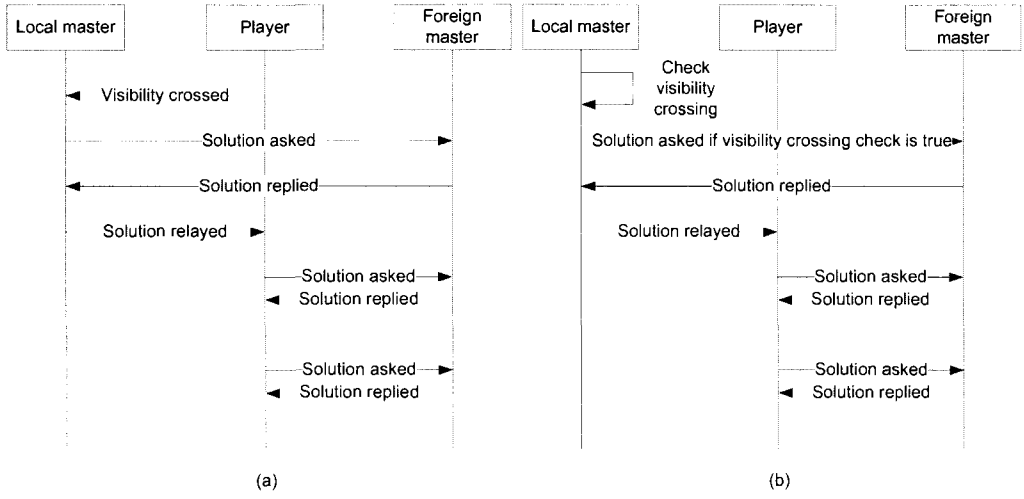

Figure 4.13: Solution to visibility problem: (a) a simplest approach (b) a smarter approach

asking the foreign master to involve the player in the exchange of messages relevant to the player and happening in the foreign zone. But the idea of an 'explicit' message is redundant since the master is already a member of the player's ALM tree and so this can be automated as follows. Whenever the master receives a message from a player, it checks whether the players' visibility exceeds the border of the zone (Figure 4.13b).

If so, it means the player needs to see into the neighboring zone, and so the local master forwards a control packet to the foreign master (i.e. the master of the neighboring zone) asking for the player to be involved in future message exchanges happening in the neighboring zone. This will solve the visibility problem. At this point, and while the player still hasn't crossed the zone boundary, the message exchange is done directly between the master of the neighboring zone and the player; i.e., the player does not join any P2P clusters of the neighboring zone. If the player does cross the zone boundary, a hand-off occurs according to the design presented in

section 4.7

4.7 Seamless Player Hand-Off

Despite *entity typing* and smart *interest-driven zone crossing* with *buffer regions*, zone transition may not be seamless. When a player crosses a zone, the player's area of interest can overlap with one or more zones. So, inter-server communication becomes necessary as discussed in the previous section. But, in addition, there is a possibility of interrupted communication when players cross zone boundaries as the P2P overlay network is re-tuned. One of the straightforward techniques to address this problem is the sharing of game states among zone masters; i.e, each zone master not only knows the game state of its own zone, but also that of its surrounding zones. Using this approach, if a player crosses into another zone, the master of the new zone has the game states of both the new zone and the old zone of the player, and can therefore still provide the player with what's happening in the old zone while adding the player to the new zone, causing no perceived discontinuity for the player. While this nave approach can achieve sufficient fault tolerance and proper consistency control, it has a couple of problems. First, it is an impractical solution as it requires sharing of a huge amount of information (six neighboring zones for each master plus its own zone), and might lead to overloading the masters. Second, even if masters somehow have this huge capacity, it is a wasteful approach in terms of bandwidth as most of the messages from neighboring zones have no effect in the gameplay of the current zone.

A better approach would be to share only the relevant state information with adjacent zones. Instead of sharing the complete state information of a zone, we propose state sharing of only the buffer regions. A buffer region is defined as an area between two adjacent zones which belongs to both zones, as shown in Figure 4.14. The region marked by 1/2 and 2/1 is a buffer region; hence, zone masters 1 and 2 always share states of the players that fall inside those buffer regions. This reduces

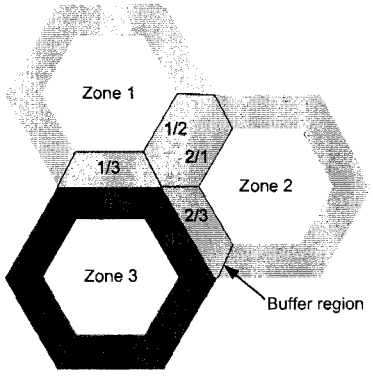

Figure 4.14: State sharing leads to smooth zone crossing

Table 4.2: Message exchange reduction for different buffer sizes

Size of 1/2	**Number of enclosed zones**						**% of msg. exchange**
buffer zone	**1**	**2**	**3**	**4**	**5**	**6**	**reduction per zone**
1/8	0.13	0.25	0.38	0.50	0.63	0.75	87.50
1/9	0.11	0.22	0.33	0.44	0.56	0.67	88.89
1/10	0.10	0.20	0.30	0.40	0.50	0.60	90.00
1/11	0.09	0.18	0.27	0.36	0.45	0.55	90.91
1/12	0.08	0.17	0.25	0.33	0.42	0.50	91.67
1/13	0.08	0.15	0.23	0.31	0.38	0.46	92.31
1/14	0.07	0.14	0.21	0.29	0.36	0.43	92.86
1/15	0.07	0.13	0.20	0.27	0.33	0.40	93.33
1/16	0.06	0.13	0.19	0.25	0.31	0.38	93.75

the exchange of the number of messages between the two zones. However, the core zones 1, 2 and 3 are solely covered by the zone masters 1, 2 and 3, respectively.

If we assume that the message exchange rate is proportional to zone size, we can see in table 4.2 reduction in shared messages with respect to the number of enclosing. The table shows the improvement in percentile with respect to full-zone state replication. Thus, depending of the size of the buffer zone, approximately 90% of message redundancies can be avoided while achieving seamless player hand-off within the buffered region.

4.8 Summary

In this chapter, we have seen the overall architecture of MM-VISA and its various components, each of which helps resolve a specific problem that is encountered in zonal MMOGs. We saw clustering of players in a zone can stabilize the P2P overlay network. Each zone has a zone master responsible to build a P2P overlay network for intra-zonal communication among players. Interest-driven zone crossing and dynamic shared regions between adjacent zones were also proposed that can take informed decisions about zone-crossing and reduce the number of overlay switching operations. Moreover, the message overhead reduction policy described can reduce consumption of bandwidth. It was also shown that through game state sharing among the masters, seamless zone switching can be achieved increasing the quality of experience perceived by players. But, all of these operations and duties will lead to server load and so load balancing becomes important since a server could potentially be overwhelmed if it has too many players. In the next chapter, we will discuss in details how load balancing can be used to alleviate this problem.

Chapter 5

Hotspots and Load-Balancing Mechanisms

In this chapter, the term 'Load' means how many players or objects are waiting in the queue to access the computer resource. In other words, the system load is a measure of the amount of work that a computer system can perform. This is indeed calculated for a certain period of time. As explained earlier, the proposed hybrid MMOG model divides the virtual world into several manageable zones where each zone covers the players in a close proximity. Each zone has a zone master that builds a P2P overlay network for intra-zonal communication among players. Collectively, the set of zone masters forms a top-level management mechanism that regulates the operation of the MMOG. MMOG applications require much network bandwidth to function properly, and so in a distributed MMOG server architecture, the server nodes can become overloaded by the high number of players with their generated packets. Thus, a load balancing algorithm becomes necessary which distributes the inbound traffic to multiple servers.

In MMOGs, it is possible that many players can move into a zone, degrading quality and affecting players' gaming experience, which is undesirable. This problem is

known as the hotspot problem. On the other hand, load balancing can be defined as a scheme to keep servers' load at a predefined level despite the presence of hotspots while retaining desired game service. At its simplest, a load-balancing algorithm attempts to assign equal numbers of players to partitions while minimizing communication costs between partitions. A partition's sub-zone consists of the data exclusively assigned to it. The union of sub-zones is equal to the entire problem domain. But the workload in a continuously changing system evolves over time, so a partitioning policy that works well for a static problem or for a slowly changing problem may not be efficient in a highly dynamic situation. In this chapter, three new load-balancing algorithms will be presented in the context of zonal MMOGs.

5.1 Load Balancing using Two Level Partitioning

Hotspots affect the quality of gaming where event response times become unacceptable. There is a need to define *Service Level Agreements* (SLAs) for such scenarios: what is the tolerable limit? Chen et al. assume one second for load balancing and two seconds for aggregation (algorithm used to restore locality) as an acceptable delay for load management to stabilize [CWD+05]. A slight variation is sometimes acceptable.

5.1.1 On-the-Fly Two-Layer Partitioning

The basic idea is that if we increase the granularity of zones, we ultimately decrease the zone sizes. By decreasing the size of each zone, one reduces the probability of having a high number of players in a zone. Thus, there will be fewer players in a zone, and thereby reduced load. Consider Figure 5.1, where Zone 1 is marked as a hotspot. Say the master managing Zone 1 detects the hotspot (explained later). It then applies the load-balancing algorithm for this zone. Each zone in itself has a zonal partitioning also, following a fixed partitioning configuration (fixed number of

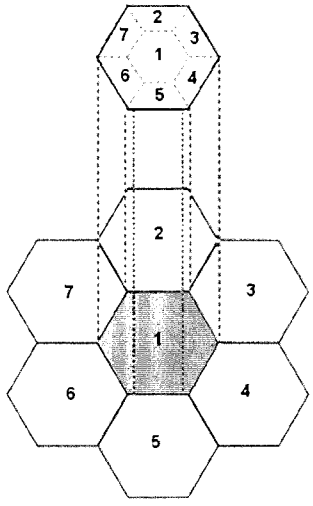

Figure 5.1: Two-layer partitioning

partitions). This is what we call a two-layer partitioning: partitioning of the map and partitioning of the individual zones. All the partitioning is done when the game is first loaded. Having a fixed configuration for the second-layer partitioning, the adjacent servers know which partitions they are responsible for in case of overload. When a load-balancing mode is triggered from Zone 1, zone masters 2–7 can quickly take responsibility for managing nodes inside the partitions. Having this pattern in partitioning is the key point of design contribution: everything is set for hotspot scenarios. The only dynamic element incorporated is the hotspot zone triggering the load-balancing mode, and nodes subsequently migrating to their new servers. Since a big part of the hotspot is now managed by the adjacent zones, the load will be dispatched among six adjacent zone servers. Furthermore, the locality is preserved: only adjacent zones accept the shed load, so there will be no need for too many node-specific connections.

5.1.2 Rejecting a Load

One of the problems introduced by such architecture is the case when an adjacent server cannot accept a shared load from the overloaded server. Maybe the adjacent servers themselves are experiencing hotspots and cannot accept additional load. If a zone is a hotspot, and one of its adjacent zones triggers a load-balancing mechanism for itself, it simply rejects the request and will not assume responsibility for its part.

It is important to discuss to what extent shedding of load to adjacent servers disrupts these servers. First, the adjacent servers are only sharing a small portion of the load: 1/8 of the area of the hotspot zone. Even though we cannot assume even distribution of nodes to zones, we still consider such a portion relatively small. Second, if the load shed is small but creates an overload when added to the original load, then the zone becomes a hotspot itself and can trigger load shedding to its adjacent zones. This might create a domino effect, but at some point, it will settle on having adjacent zones without large existing loads taking up the slack. Finally, a direct result of these hotspots is having zones that are partially or completely empty. So we can say that an overload in one zone will only cause a small disruption for adjacent zones. Moreover, such scenarios, when adjacent zones accept load and become overloaded themselves and then shed load to their adjacent zones, are a good picture of how hotspots should be managed. The original zone configuration will change and converge to manage the hotspot, the center of attention of the game or virtual environment at the moment.

5.1.3 Multiple Hotspots

A single-hotspot scenario is fairly simple to deal with as discussed earlier. A multiple-hotspot scenario, on the other hand, becomes complicated, and it is important to explain how the architecture works in this situation. Consider Figure 5.2; Zones 5, 6 and 9 represent hotspots. The order in which each zone became a hotspot is as follows: Zone 5, 6, then 9. When Zone 5 becomes a hotspot, each of its adjacent mas-

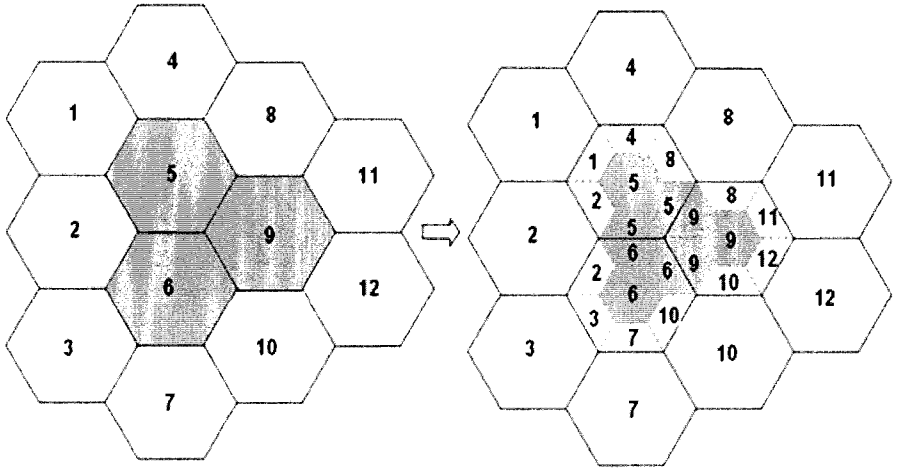

Figure 5.2: Multiple-hotspot scenario

ters assumes responsibility for managing its partition, as explained in the previous section, including master 6 and 9. When Zone 6 becomes a hotspot itself; however, it abandons managing its partition in Zone 5, and master 5 resumes management of the partition adjacent to Zone 6. The same case applies when Zone 9 becomes a hotspot: Master 9 abandons managing its partitions in zones 5 and 6, and masters 5 and 6 resume management of the partitions adjacent to Zone 9. Figure 5.2 clearly shows the end result of the server to partition mapping in a multiple hotspot condition. One can see that the overloaded masters are now managing smaller areas, and how the original partitioning configuration adapts to the multiple-hotspot situation. Notice how the original partitioning configuration converges into the second-layer partitioning, keeping the same uniform shape. This regularity in partitioning is a critical point of the architecture and an important result of hexagonal zoning.

5.1.4 Undo Mode

It is intuitive that there should be a mode in which the original partitioning is restored once a hotspot ceases to exist. This is important to preserve the regularity of the first-layer partitioning and reassign servers to zones evenly. We propose that once the hybrid server is back to less than 70% of its overload threshold, it trigger an undo mode, so that nodes can migrate back to their original hybrid servers. The 30% margin is given to prevent oscillations: it makes sure that the hotspot case will not recur within a short time. (The 30% is a suggested value. This percentage can be changed depending on implementation and performance analysis of the undo mode.) Since we cannot assume global knowledge of the load, hybrid servers frequently share lists of the nodes they are managing (position of nodes, which zone they belong to, etc). When a hybrid server triggers a load-balancing mechanism after its first-layer partition becomes a hotspot, it periodically checks the cost of managing the other second-layer partitions (restoring its original first-layer partition). If, after adding this cost, the total load is less than 70%, it triggers the undo mode.

5.1.5 Discussion

Lu et al. suggest that there is a contradiction in the trend of research concerned with dynamic load management in MMOGs [LPM06]. On one hand, server-side inter-communication is minimized to promote scalability and balance load. On the other hand, inter-server communications are relied on to alleviate process exhaustion due to crowding. It is very important, therefore, to choose an appropriate zone size to minimize inter-server communication, or node-specific connections in this case. Tumbde et al. also argue that the Voronoi partitioning can lead to the formation of very small zones, in case of clustering of nodes [TV04]. This can lead to formation of zones which are smaller than AoI of a player. Such a small fragmentation is unnecessary and may lead to additional communication overhead. For these reasons,

zone sizes should surpass nodes' visibility or interaction ranges, so that inter-zone communication is temporary and minimal. When choosing the sizes for the our layered partitions, we try to balance the process of adjacent servers taking part in the load in the hotspot area, and preserving locality and isolation of nodes in one zone from those in another to reduce inter-server communication. This promotes message isolation and keeps interest well managed.

Another problem that our architecture has to deal with is the migrations of nodes from the original server managing the hotspot to the adjacent servers. Such a problem is considered a natural result of load balancing. Still, a predictive approach can be considered in triggering the load-balancing mode before it is needed to avoid accepting nodes in the first place and reduce migrations.

The architecture presented achieves four objectives: simplicity, practicality, flexible configuration, and preservation of the ALM-based tree architecture. At the time of load-balancing, a set of players, based on the algorithm, is moved from one server to another. This definitely requires a maintenance operation in the ALM trees. Instead of constructing a new P2P overlay, we perform a repair operation which keeps the maintenance cost at a lower value and preserves the ALM in some sense. No additional third-party processors are added, such as schedulers that get to decide who accepts the load in an overload scenario. The architecture presented is only dynamic in switching modes: zoning is layered and only the switch between the zonal information is dynamic. Locality is preserved when only adjacent zones assume responsibility for a hotspot. So, in contrast to other complicated load-management algorithms, this technique is simple and can achieve desirable results in real time.

5.2 Load Balancing: Multilevel Multiphase approach

The *Multilevel Multiphase Load Balancing* (MMLB) method presented in this thesis is designed for fixed-size zones. It works in three phases and can be employed according

to the load scale in order or independently. The first phase works for the top level of microcells/zones and takes care of the inter-server communication while regulating the load among the servers. The second phase works in a preventive manner and reduces load by discarding state-sharing policy with the neighboring microcells/zones. The third phase decomposes the top-level microcells into the deep-level microcells and sheds load with the help of enclosing zone masters. This new MMLB technique limits structural reformation penalties and gives zone masters, i.e. servers, a provision to reduce load in a step-by-step manner. This avoids bouncing back and forth from the top-level partitioning to the deep-level partitioning and reduces control overhead to a significant extent.

5.2.1 New Load Definition

The first step to deal with a hotspot is to detect a loaded zone. Most of the existing approaches consider the total number of players in a zone to define the load. If the total number of players in a zone exceeds the maximum affordable limit, it is a hotspot. However, we believe looking at the question in a different way would be more appropriate. In an MMOG, there are many players with different attributes like soldiers, robots, aircrafts, etc. These behave differently, so the total number of entities in a zone is not necessarily a good indication of the load. Also, actions and events do not depend on the number of players. For example, when a building explodes, it sends update messages to all players in the vicinity. This update generation does not depend on how many players are around. In fact, a zone with a smaller number of players but with more actions can be a hotspot, in contrast to a 'peaceful' zone with a much larger number of players. Thus, the load is not directly related to the number of players and, in fact, is linked strictly with the message generation rate. Thus, we believe message generation rate, $R_m(t)$, is a better way to define the load of a zone.

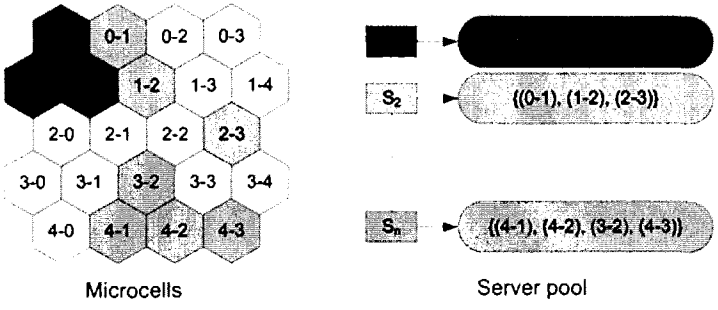

Figure 5.3: The layout of microcells and a random microcell distribution to servers

5.2.2 Identifying the Loaded Server

Let a set of microcells/zones define the virtual world that is maintained and coordinated by a set of servers called the server pool. The server pool with n servers is represented by the set $S = S_1, S_2 \ldots S_n$. Each server, e.g. S_i, serves one or more microcells depending on load. The microcells are given unique numbers in left-to-right and top-to-bottom order. The layout of microcells and a random microcell distribution to servers are shown in Figure 5.3. Say the server S_i is in charge of a set of microcells $M_i = \{m_{i_1}, m_{i_2} \ldots m_{i_k}\}$. Let $r_{i_j}$is be the message-generation rate of the microcell m_{i_j}. Thus, the load of the server S_i according to message generation rate is:

$$R_m(S_i) = \sum_{j=1}^{k} r_{i_j} \tag{5.2.1}$$

Let us define a message threshold term T_m. Thus, a server is loaded or there is a hotspot if $R_m(S_i) \geq T_m$. $R_m(S_i)$ can be computed and refreshed by exploiting the total packets processed over a period of time (say, every minute). A good recommendation for T_m could be to take the average of $R_m(S_i)$ of the entire game space and increase it by p%,

$$A_m = \sum_{i=1}^{n} R_m(S_i)/n \tag{5.2.2}$$

$$T_m = A_m + p\% \times A_m \tag{5.2.3}$$

5.2.3 Phase 1: Load Balancing for Top-level Microcells

When there is a hotspot, the load of the concerned server is relieved by moving one or more microcells to other, less heavily loaded servers. As the movement of microcells introduces complexities, some intelligence is required to identify the best possible set of microcells that can be moved. The reduced deployment and maintenance overhead of hybrid MMOGs comes at the cost of performance due to peer dynamics. Many techniques have been proposed, such as entity typing and smart interest-driven zone-crossing with a buffer region to maintain the desired level of performance. As it is difficult to predict avatars' movement around a cell boundary (possible frequent in-and-out movements), it could be effective to share the relevant state information between both microcells as explained earlier.

The objective of changing a microcell's ownership from one master to another is to approach a hotspot. The key point that must be taken care of is to minimize the inter-server communication while carrying out such a process. Let I_i be the number of communication links among avatars completely internal to the microcell i and $B_{i,j}$ be the number of communication links involved between microcell i and microcell j in the buffer region. Let the server S_i be overloaded and currently serving k microcells. The target is to determine an ordered list of microcells so that changing the ownership of microcells in that order introduces less performance penalty in the overlays.

The algorithm initially forms a group of microcells based on the microcell's neighborhood properties. A microcell can be a member of a nonempty group if and only if it has a common edge with any microcells of that group. Assume that after this grouping policy, Server S_i has l groups $G_1^i, G_2^i \ldots G_l^i$ in order of their cardinality, i.e. $|G_1^i| \leq |G_2^i| \ldots \leq |G_l^i|$. The general strategy is to find a microcell within a group with a minimum number of communication/interested links among the microcells in that

group. Due to the change of ownership, these links become foreign links that cause inter-server communication. Thus, a group with more microcells could have more common edges than a small group, at least heuristically. Thus, it can be effective to keep a large group intact. Thus, a group with a fewer microcells is chosen for load balancing and so the system chooses the group G_1^i. Finally within a group, a microcell with a smaller number of potential foreign links, i.e. $B_{i,j}$, is chosen for handover to the less-loaded server, which is identified by equation 5.2.1.

5.2.4 Phase 2: Exploiting Buffer Region to Release Load

In the case of a hexagonal zone layout, generally, a zone has six other enclosing zones. A buffer region is defined between two adjacent zones to provide seamless player hand-off (as explained in the last chapter). We can take advantage of this buffer region to reduce a master's load when it reaches its maximum. First, we keep track of the load of the enclosing zone masters in terms of $R_m(S_i)$. The load-balancing procedure can be activated when $R_m(S_i) \geq T_m$. In this second phase, each neighboring zone master can reduce a fixed amount of load from the hotspot, up to six times (six neighbors) if needed. The order of the cooperating masters will be the increasing order of $R_m(S_i)$ of the enclosing masters. One advantage of having this phase is that when zone master x stops sharing the states of buffer region with zone master y, it actually releases load from both zone masters with respect to that buffer region. Naturally, the magnitude of released load depends on the size of the buffer region. Phase two of the MMLB removes replicated information used for seamless player hand-off to reduce the load. However, when there is a hotspot, removing the state-sharing feature is not without value.

Table 5.1 summarizes the released load at each step for different buffer sizes. The table covers fictitious data. The motivation of this presentation is to show the applicability and significance of load releasing with respect to the buffer region. For

Table 5.1: Load release at different stages for various buffer sizes against current load

	% of reduced load for different buffer sizes								
Step	**1/8**	**1/9**	**1/10**	**1/11**	**1/12**	**1/13**	**1/14**	**1/15**	**1/16**
1	7.14	6.67	6.25	5.88	5.56	5.26	5.00	4.76	4.55
2	7.69	7.14	6.67	6.25	5.88	5.56	5.26	5.00	4.76
3	8.33	7.69	7.14	6.67	6.25	5.88	5.56	5.26	5.00
4	9.09	8.33	7.69	7.14	6.67	6.25	5.88	5.56	5.26
5	10.00	9.09	8.33	7.69	7.14	6.67	6.25	5.88	5.56
6	11.11	10.00	9.09	8.33	7.69	7.14	6.67	6.25	5.88

example, in Table 5.1, if the size of the half-buffer region is 1/16 of a zone, it can reduce the load by 4.55% in the first step, 4.76% in the second step, 5% in the third step and so on. However, despite the approach presented here, there is still a possibility that the zone master will not be able to carry the load due to overcrowding (let us call this phenomenon an extra-hotspot). In that case, the third phase can be applied.

5.2.5 Phase 3: Load Handling through Deep-level Partitioning

An extra-hotspot is not unusual even after the use of phases one and two described above. In this situation, the overcrowded zone master cannot handle its load and this requires a move to the third phase. The general simulation layout is shown in Figure 5.4a. When the current load exceeds what can be afforded, i.e. $R_m(S_i) \geq T_m$, the zone master switches to the third phase, i.e. deep-level partitioning. In this condition, a

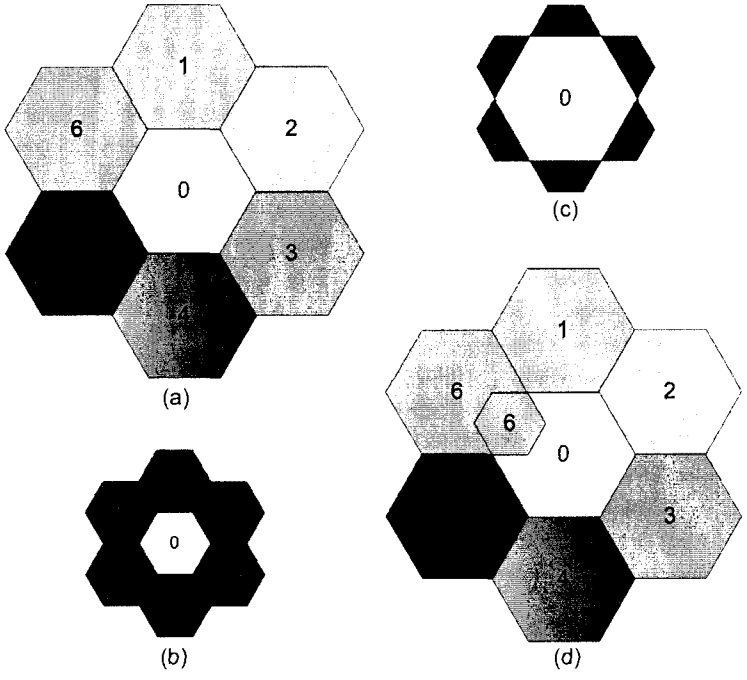

Figure 5.4: Stepwise deep-level partitioning for load balancing

zone is decomposed into seven other, smaller hexagons (deep-level microcells): one is completely inside the current zone and other six small hexagons are overlapped with six enclosing zones (Figure 5.4b). Thus, in the worst case, a zone master must partially shed the load of the six enclosing zones. In that case, its coverage is identical to Figure 5.4c. The key difference from the existing load-balancing methods is that when the system is loaded, the existing techniques immediately involve all the surrounding zone masters. This significantly reduces load in the extra-hotspot area, but it is not actually necessary to involve all the zone masters. Indeed, this unintelligent one-step switching adversely affects all the enclosing zone masters. Similarly, the undo operation is carried out in a single step. Thus, the zone master in the center treats the load at the two extremes: hotspot, followed by a very low load, which soon can become a heavy load due to the undo operation, and so on. To overcome this, here we

Table 5.2: Load shedding comparison

	% of load shedding against current load	
Step	**Current approach**	**Generic approached [KAS07]**
1	12.50	75.00
2	14.29	not application
3	16.67	not application
4	20.00	not application
5	25.00	not application
6	33.33	not application

choose a step-by-step load-shedding mechanism as needed. Thus, for example, when zone 0 is marked as an extra-hotspot, zone master 6 offers its support (Figure 5.4d). If the support of zone master 6 is insufficient, another zone master (such as 5) can be involved, and so on, until the load reaches below the threshold T_m. Similarly, the undo operation is carried out in steps.

This strategy reduces switching back and forth between the two levels of partitioning, which is more appropriate for a hybrid architecture where overlay stability is more important. Due to this stepwise load-shedding mechanism, the hotspots are less likely to propagate and bounce back and forth across different zones. Table 5.2 summarizes the effectiveness of the stepwise deep-level partitioning compared to the generic approach presented in the earlier section. The numerical value depends on how many players are inside each deep-level microcell, and here in the table we have considered a simple case of homogeneous distribution of players across the microcells.

5.3 Load Balancing for Non-Uniform Zones

A notable point, in zonal MMOGs, is the fixed zone size. There are a few approaches that deal with dynamic zone shaping, e.g. *Voronoi diagrams*, but most of these designs do not have load-handling mechanisms. It is difficult to predict player density or distribution before the start of a game. As a consequence, if a server is pre-assigned to a zone, it cannot share its resources with surrounding zone servers. The problem persists even when the server pool should be able to cope with the total load, at least theoretically. In this section, we present a load-balancing method where zone shapes are not predefined which can be adjusted with respect to load. It should be noted that we cannot intermix the earlier approaches with this one. It is completely a different method.

5.3.1 The Algorithm

Consider a system with N masters/servers with a big single zone, i.e. the entire virtual world, where players join and leave over time and initially only one server is involved. In this adaptive zone-shaping approach, the game field is partitioned dynamically based on players' logical position and interaction patterns. Thus the resultant zone shapes are not uniform. The bisection procedure is used to partition a zone into two sub-zones of nearly equal load while attempting to minimize the communication cost, i.e. the number of links crossing logical boundaries. According to the bisection procedure, the zone is first cut into one dimension to yield two sub-zones. Then other cuts are made repeatedly in the new sub-zones whenever needed. The simplest form of the recursive bisection procedure is the 'coordinate bisection', which is generally applied to irregular grids that have a local communication structure. This technique makes a cut, i.e. boundary, based on the physical coordinates of a region (logical position in our case). At each step, it subdivides along the longer dimension so that if the cut is made along the X– dimension, the grid points in one sub-zone will

all have an X– dimension lower than the grid points in the other. This technique has the advantages of being simple and inexpensive, but it does not consider inter-zone communication between two sub-zones while partitioning. Here, we present a modified approach that considers this important parameter.

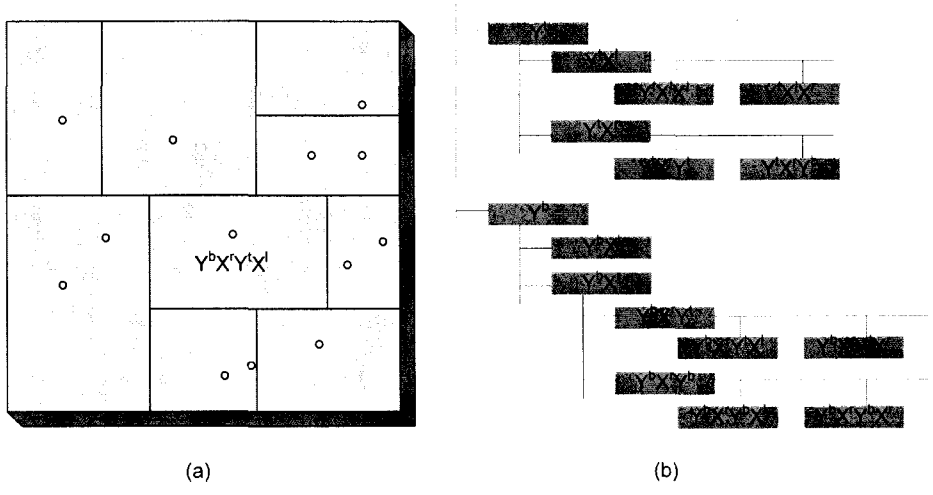

Figure 5.5: (a) Virtual world partitioning (b) Zone naming

In our approach, the bisection along the X– dimension generates two zones marked X^l and X^r. Similarly, the bisection along the Y– dimension yields Y^t and Y^b. Thus, a zone can be identified through the sequence of such marks. For example, as can be seen in Figure 5.5, the zone $Y^bX^rY^tX^l$ is the outcome of bisections along Y (bottom), X (right), Y (top), and X (left) dimensions, respectively. It should be noted that this is not merely an equal bisection of a zone in terms of size but in terms of communication cost. First, it orders all players based on either X or Y dimension. Say $< P_1, P_2 \ldots P_n/2 \ldots P_n >$ is an ordering of players according to the X dimension. The algorithm initially partitions players in two halves, X^l and X^r, where players $< P_1, P_2 \ldots P_{n/2} >$ and $< P_{n/2+1}, P_{n/2+2} \ldots P_n >$ belong to X^l and X^r, respectively. In literature, there are several approaches to estimating load in a zone. The simplest

one is the number of players in a zone. The presented bisection technique belongs to this category.

This coarse partitioning can be improved by moving the cut along the vectors perpendicular to the boundary in either direction. The target of the cut movement is to reduce the *cut size*, which is the number of edges crossing the partition. In order to be effective, we restrict the cut movement by length L in both directions, as shown in Figure 5.6. Let A and B represent the set of players in their respective cut scrolling zones, i.e. Z_A and Z_B. We sort the players of both zones by increasing distance from the cut. The first players (a and b) from each ordered set are picked where $a \in A$ and $b \in B$. The internal and external costs are calculated for the player $a \in A$ with respect to sets A and B respectively. The internal cost calculates the total number of direct connections that exist between that player and other players in the same set, $I_a = \sum_{v \in A} C_{a,v}$. The external cost calculates the external connections associated with that player, i.e. $E_a = \sum_{v \in B} C_{a,v}$. The term $C_{a,v}$ is 1 if there is an interested link between player a and player v, ; otherwise it is 0. Then the cost difference is calculated, i.e. $D_a = E_a - I_a$. Let $D_{a \in A}$ and $D_{b \in B}$ be determined by the algorithm presented here. Say $D_{a \in A}$ is negative. This means that player a has more internal interested links, so we do not need to move the cut. But if $D_{a \in A}$ is positive, this means that player a has more interested links in Z_B. It also indicates that placing player a in set B is more effective. Ergo, we need to move the cut to the left to keep them in the same group B. The cut movement is controlled by these values and is given in Table 5.3. The algorithm continues until there is no feasible cut scrolling area or it satisfies rule 1, as shown in Table 5.3. In this way, the algorithm partitions the virtual space over time based on the number of players and their logical positions. It should be noted that the system can provide this game service as long as the total resource is sufficient for the whole population irrespective of the players' logical position. Thus, it can handle hotspots.

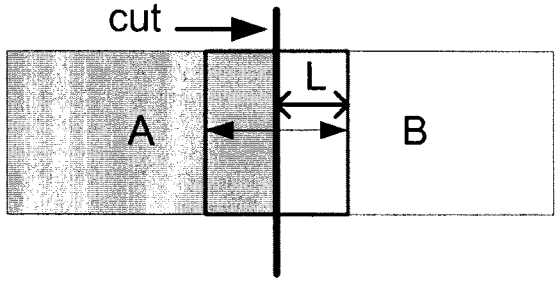

Figure 5.6: The cut movement

Table 5.3: The cut movement rules

Rule no	$D_{a\in A}$	$D_{b\in B}$	Decision
1	-ve	-ve	Keep the cut as it is
2	-ve	+ve	Move the cut in the direction of Z_B
3	+ve	-ve	Move the cut in the direction of Z_A
4	+ve	+ve	Two options a) Keep the cut as it is b) Move the cut based on the higher difference

5.3.2 Zone Merging

Naturally, an overloaded place in one part of a virtual world can create a low-population space somewhere else. Thus, there is a scope of merging of unloaded zones. In this regard, the following policy can be applied to merge two zones, Z_A and Z_B, if:

1. One of the corresponding zone masters is capable of handling the total load
2. The lengths of the zone names are equal– say l is the length of the zone names, and

3. $\texttt{prefix}(Z_A) = \texttt{prefix}(Z_B)$, and both prefix have length $l - 1$

Condition 1 checks the feasibility of zone merging while condition 2 and condition 3 keep the regularity of zone shapes, i.e. rectangles.

5.3.3 Zone Capturing

Sometimes, zone merging is not feasible, but the load of a particular zone is small enough that it could be handled by another master. If so, such a master is discovered with respect to the current load distribution. Hence, a master can be the coordinator of multiple zones, and 'captures' other zones. It is imperative that zone-merging preserves the locality of player interactions. Thus, zone-merging is more effective than zone capturing and should be applied whenever possible.

5.3.4 Dynamic Cut Movement

In order to make load sharing more countable, a dynamic cut movement can be used. The frequency of this attempt depends on the load on both sides of the cut. The cut movement toward a zone is also a function of the unused resources of that zone.

5.4 Summary

In this chapter, I have proposed new load-balancing methods for multiplayer games. The issue addressed here is how to deal with hotspots. The steps are: 1) measure message exchange rate; 2) determine hotspots; and 3) apply a load-balancing algorithm. All presented approaches try to evenly distribute load across different zones formed over the game map. The presented MMLB works in three phases and can be used according to load scale; in order or independently. The first phase works for the top level of microcells/zones and takes care of inter-server communication. The second

phase works in a preventive manner and reduces load by discarding state-sharing policy. The third phase partitions the top-level microcells into the deep-level microcells and sheds load with the help of enclosing zone masters. The advantage of the presented MMLB is its low structural restoration at the time of load balancing. This is accomplished with the help of a layered zone layout. The new MMLB technique gives masters a provision to release load in a step-by-step manner that eventually avoids bouncing back and forth from the top-level partitioning to the deep-level partitioning and reduces control overhead to a significant extent. This chapter also presents an adaptive zone shaping mechanism using a bisection procedure that does not stick to any predefined fixed zone size. The modified bisection procedure applied here finds a cut that can efficiently partition a zone considering both the population size and players' area of interest.

Chapter 6

Quality and Design Issues

There are several ways to improve the quality of MMOGs. This chapter covers three new approaches – dynamic area of interest management, expedited state dissemination, and a time constraint compliance algorithm for improved MMOG systems. Broadcasting all update messages to every player is not a viable solution to maintain a consistent game world. To successfully overcome this challenge, multiplayer online games need to employ sophisticated interest management techniques. In Section 6.1, we examine some of the most popular area of interest management (AoIM) techniques for online games, and propose a new dynamic AoIM method suitable for peer-to-peer architectures to characterize the interaction space in real time. In Section 6.2, a novel state-dissemination model for hybrid MMOGs is presented by exploiting idle periods of the participating players. This new approach can perform such tasks much faster than other traditional approaches with an identical environmental setup. It can also expedite state sharing even in heterogeneous environments. In Section 6.3, a new quality control algorithm is presented considering players' virtual and geographical positions. We assume that the interaction details between two players is inversely proportional to their virtual distance. Based on this assumption and time constraint for the target application, the gaming experience in MMOGs can be tuned.

6.1 Dynamic Area of Interest Management

Area of interest management for online games is a challenging task. In the literature, there are some effective methods for interest management, but considering a peer-to-peer gaming framework, these interest management approaches are not readily applicable. As a consequence, for the proper use of players' bandwidth and computing power, we either need to discover new interest management techniques suitable for this environment or to adapt older approaches accordingly. This section presents a dynamic interest management technique.

6.1.1 The Proposed Approach

The mapping of an area of interest to a fixed-size zone is common. In short, the unification of an AoI to a zone is straightforward as logical spaces are predefined. Thus, players staying in a zone are considered to have a common interest confined to that logical space. But, due to the nature of a game, an area of interest overlaps multiple zones and breaks the significance of zone formation. In addition, it requires regular inter-zone communications for various reasons, like solving the visibility problem. Thus, it makes a system more vulnerable to heavy load. Here, the new interest management model can evolve over time and create logical space as needed.

Initially, each player has its own area of interest, which is defined by its visibility scope. Say p_i is a player whose area of interest is defined by $aoi(p_i)$. Let's assume, two players p_i and p_j are interacting with each other. This is a symmetric relation of interest – i.e., if player p_i is interested in player p_j, then player p_j is also interested in player p_i (as shown in Figure 6.1). This symmetric relation is not always true. Sometimes there can be asymmetric relations in the case of radar, sensor arrays, different entity types, etc., but here we are considering only symmetric relationships. When more than one player has a common area of interest, then the resulting interested area can be defined as $AoI(i) = \{p_{i_1}, p_{i_2} \ldots p_{i_k}\}$ where k is the number of players. So,

a game space consists of many AoIs, and can be defined by $G_{space} = \bigcup_j AoI(j)$.

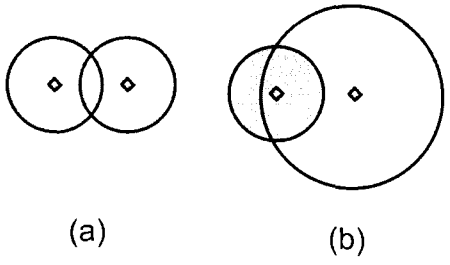

Figure 6.1: The symmetric and asymmetric relation of interest

Let us consider a hybrid MMOG framework, which is more realistic than a pure peer-to-peer architecture and more scalable than a pure centralized system. The system is administered and coordinated by a set of servers that also try to use players' resources whenever possible to relay game states to others to reduce server load. The key design characteristics of the model are:

- At the beginning of a game session, each player discovers his/her interested area through a server;
- Each AoI is monitored by a server that provides different maintenance services like tracking the AoI scope with the help of participating players (explained later);
- The overlapping conditions of AoIs are checked at regular intervals to reduce communication overhead.

Due to mobility, players' position and interaction space can change regularly. In addition, the peer-to-peer part of the architecture makes it more challenging because of players' heterogeneity in terms of bandwidth and processing power, and their widespread distribution over the Internet. Thus, an appropriate and effective area of interest (AoI) structure is important. It must keep the AoI maintenance cost to

a minimum while providing the desired level of game services. The key intuition in this regard is to discover a subset of players for each AoI and hand over AoI scope tracking responsibility to them. These responsible players can properly reshape AoIs when necessary without involving all players in the vicinity. That ultimately reduces maintenance cost. The proposed algorithm is explained bellow.

Algorithm 2 The area of interest and its scope-tracking node-discovering algorithm

Require: P: the set of players

1: $i := 0$

2: $hull[i]$:= the leftmost point of P

3: **repeat**

4: $hull[i+1]$:= a point such that all other points in P are to the right of the line $hull[i]hull[i+1]$

5: $i := i + 1$

6: **until** $hull[i] \neq hull[0]$

7: Scope-Tracking-Nodes := hull

Let an AoI have k players represented by $AoI(i) = \{p_{i_1}, p_{i_2} \ldots p_{i_k}\}$. We form a convex hull to present such an AoI. A convex hull for a set of points is the minimal convex set covering that set. The simplest algorithm in the plane was proposed by *R.A. Jarvis* in 1973, called a *gift wrapping* algorithm with a time complexity of $O(nh)$, where n is the number of points in the set, S_n, (the players, in our case), and h is the number of points in the hull responsible for tracking an AoI, in our case. The algorithm starts with a point p_0 in the convex hull. One way to discover such a point is to find the leftmost point in the set. The algorithm then selects the next point in the convex hull, say point p_{i+1}, such that all points are to the right of the line $p_i p_{i+1}$. This point may be found by comparing the angles of all points with respect to the point p_0 taken for the center of coordinates. The algorithm then starts processing with respect to the point p_{i+1}. These steps continue until it reaches p_0 again. In this

way, we define the scope of interest. Let S_h be the subset of S_n, i.e. $S_h \subseteq S_n$, that forms the convex hull. The area of interest and its scope tracking node discovering algorithm are given in algorithm 2.

The size of the convex hull changes due to the players' random movement in the game space. All players in an AoI know who is at the convex hull periphery, i.e. S_h. Thus, each player can independently determine its position with respect to the convex hull. So, if a player is inside a convex hull, it has no role in redefining the convex hull at that instant; otherwise it is a candidate to redefine it when needed. Considering the nature of the game at hand, we do not redefine the convex hull instantly if a player crosses the boundary as we do not know whether the player is returning soon. That is why we use the term *candidate*. Two attributes are incorporated to make the decision while redefining a convex hull for each candidate. These attributes are *safety-edge space* and *time-span*. A safety-edge space is defined for each edge of a convex hull, as shown in Figure 6.2. It gives a safety margin denoting the fact that the player has moved far enough and it is unlikely that it will return soon. The time-span acts like a temporal reference and is used to avoid premature decisions. Thus, these two conditions ensure a mature decision in terms of time and space. The convex hull is changed when a candidate player satisfies both conditions, initiating a task hand-off. Thus, this candidate player informs all members within the modified set of the convex hull.

The motivation of forming a convex hull is to define a confined game space for a set of players having a common interest. As the players move frequently in *random* directions, the servers require continuous tracking of players' positions to precisely detect each one's interest in the game. When a convex hull is formed for a set of players, the server can roughly discover their area of interest by monitoring and tracking those players who are defining the convex hull. The players who are inside the convex hull do not require regular interactions with the server because the peer-to-peer architecture is already managing the intra-zone communication. So, the

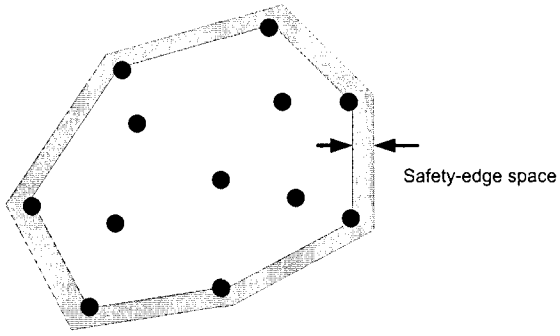

Figure 6.2: The safety margin of a convex hull

presented concept should reduce communication load of the server to a significant extent not only due to the peer-to-peer nature but also due to the confined area of interest.

6.1.2 Interest Management Classification

There are several types of interest management algorithms, which can be classified into three broad categories: proximity-based, visibility-based, and reachability-based, which are explained next.

Proximity algorithms

Proximity-based interest management algorithms are solely based on the Euclidean distance between publishers and subscribers. This type of algorithm ignores the presence of obstacles that could occlude parts of the game space. Algorithms like Euclidean Distance, Square Tiles, and Hexagonal Tiles are some examples of proximity-based interest management. The Euclidean Distance Algorithm (Figure 6.3) is purely based on the Euclidean distance among objects while the other two are approximations that use partitioning concepts. In Figure 6.3, the area of interest is shown with respect to the man at the center of the circle.

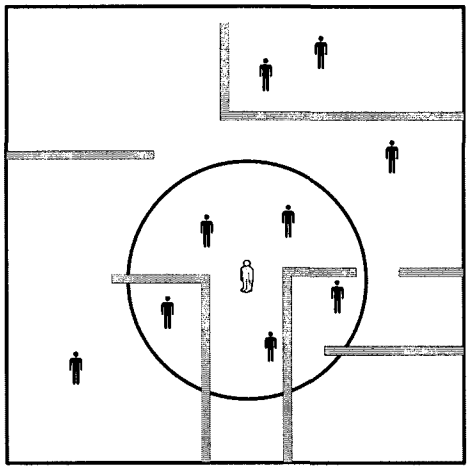

Figure 6.3: Euclidean distance algorithm for interest management

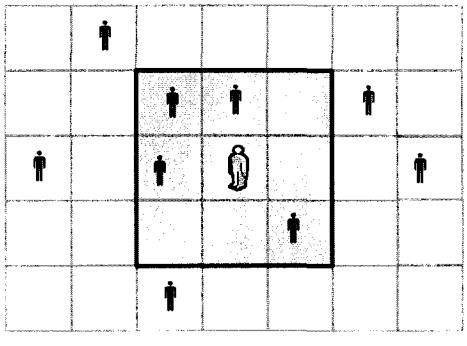

Figure 6.4: Square tile algorithm for interest management

The Square Tiles Algorithm is a simple region-based interest management where the virtual world is divided into equal-sized squares. Technically, the size of squares is set according to the radius of interest of the players. So, at any location, the subscriber is interested in at most nine tiles: the subscriber's current tile and the eight or fewer surrounding tiles (Figure 6.4). Whenever a player performs an action, the action is shared among all players subscribed to the square where the action has taken place.

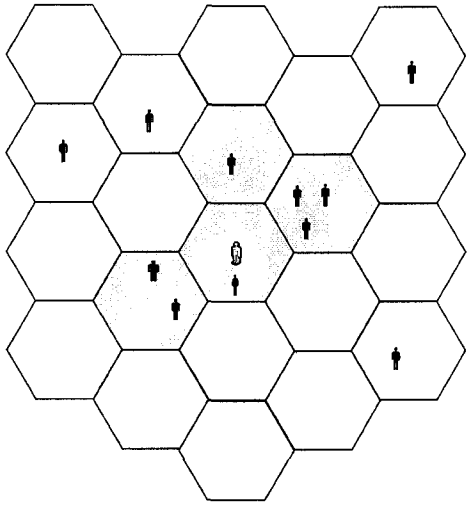

Figure 6.5: Hexagonal tile algorithm for interest management

Like the Square Tiles algorithm, the Hexagonal Tiles algorithm partitions the virtual world into equal-sized hexagons. If a player's radius of interest intersects a tile, the player subscribes to objects in the tiles. So, at any location, the subscriber is interested in at most seven tiles: the subscriber's current tile and the six or fewer surrounding tiles (Figure 6.5). For each subscriber, the algorithm performs a search from its current tile to find all tiles based on the subscriber's radius of interest. The player subscribes to all publishers contained within those tiles. This algorithm could nicely be implemented using a depth-first search, and is perhaps the most commonly

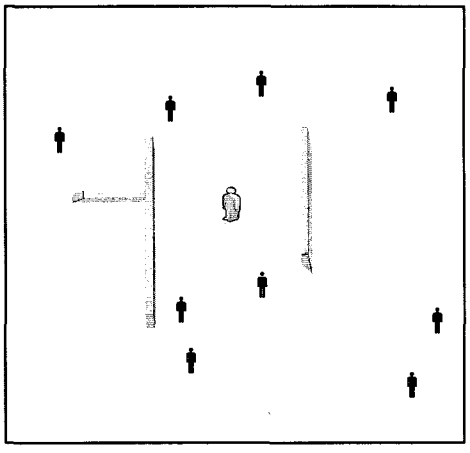

Figure 6.6: Ray visibility algorithm for interest management

used proximity-based algorithm for virtual environments and games.

Visibility algorithms

Visibility-based algorithms consider the occlusion created by obstacles in the virtual world. Theoretically, the area of interest is only limited to the player's visibility scope. In visibility-based algorithms, if a player is out of sight of another player, they are not in the same interest group even if they are physically close. Ray Visibility and Tile Visibility are two examples of this class. Ray Visibility computes the exact visibility between two objects; on the other hand, Tile Visibility uses approximation to compute the visibility between static regions. In Ray Visibility, the area of interest is uncovered with respect to the players' visibility scope (Figure 6.6). To determine an object is visible to a player, it traces a line from the position of the player to the position of the object. If the line does not intersect with any obstacle in the world, they are visible to each other. Ray Visibility is a precise interest management algorithm as it accurately determines the area of interest of a player. Its main advantage is that it provides a lower bound on the number of messages that need to be exchanged between players.

The Tile Visibility algorithm is based on the visibility between tiles. The algorithm pre-computes the visibility relationship between each pair of tiles, and the area of interest is projected after the tile visibility for each tile has been pre-computed. A player's area-of-interest is the set of tiles visible from the tile it currently occupies.

Reachability algorithms

Reachability-based algorithms define area of interest with respect to reachability even though one or more regions are out of sight due to obstacles. It is somewhat similar to proximity-based algorithms, but it discards objects that are unreachable. Unlike visibility-based algorithms, an object that is not visible (e.g., behind an obstacle) may be in the area of interest if there is a path to reach that object within its radius of interest. Reachability-based interest management approaches might be less accurate, but have the advantages that they "prefetch" information that is very likely to be relevant to a player in a near future. Tile Distance and Tile Neighbor algorithms fall into this category. A Tile Distance algorithm uses Euclidean distance between a player and a triangular tile. It runs a breadth-first search (BFS) algorithm from the current tile to discover the set of connected tiles that intersect the player's radius of interest. Then it discards tiles that are not reachable within the player's area of interest. On the other hand, Tile Neighbor algorithm defines area of interest using tile neighbor relationships. The algorithm implements a breadth-first search from the current tile of the player and determines all reachable tiles until it reaches a predefined depth. The depth is defined as a number. For example, if depth is one, the set of tiles would be all direct neighbors of the current tile.

6.1.3 Comparison

We have explained some interest management approaches suitable for client-server architecture. Each has its own advantages and disadvantages. Most importantly,

each was designed for one particular target application. But all of them are best suited for a client-server framework and not readily applicable for P2P-based online games. A categorical comparison of various interest management techniques is given in Table 6.1 which is similatr to the one presented in [BKV06].

Table 6.1: Comparison of different interest management algorithms

Euclidean Distance - proximity based and Centralized
POSITIVE: Easy to implement; Computationally inexpensive; No partitions of the virtual world
NEGATIVE: High complexity; Less realistic as it does not consider obstacles
Hexagonal Tiles - proximity based and centralized
POSITIVE: Easy to implement; Computationally inexpensive; A good benchmark
NEGATIVE: High complexity; Less realistic as it does not consider obstacles
Ray Visibility - visibility based and centralized
POSITIVE: Accurately determines area of interest; Exchange minimum number of messages between players; Efficient
NEGATIVE: Hard to implement; Computationally expensive
Tile Visibility - visibility based and centralized
POSITIVE: Simple as tile visibility relationships are pre-computed; More realistic than proximity-based
NEGATIVE: Dynamic zone shaping is not possible; Requires supporting algorithms to handle obstacles

Tile Distance - reachability based and centralized
POSITIVE: Includes some advantages of proximity-based and visibility-based approaches; Smarter than other two approaches
NEGATIVE: Computational cost is high
Tile Neighbor - reachability based and centralized
POSITIVE: Inexpensive compared to tile distance approach; Can be tuned based on applications
NEGATIVE: Less accurate compared to tile distance approach
Dynamic AoIM - distributed
POSITIVE: Good for shared domains; Maintenance overhead is distributed; Dynamic tuning of area of interest; Frequent AoI alteration and correction are controlled by time and space parameters; Can be combined with any type of proximity-based, visibility-based and reachability-based approaches
NEGATIVE: Not suitable for centralized architecture; In frequent dynamic environments performance is likely to drop

6.2 Expedited State Sharing

The fundamental difference between an online game and a streaming system is the traffic pattern and timing constraints of the transmitted data. In an MMOG, the data is represented by states that carry useful information describing a player in the virtual

world, and sharing of such states with other interested players is very important. Moreover, an event generated by a player can produce more events because of the reactive actions of other players. Thus, the traffic pattern is completely different from a streaming system. In an MMOG, the traffic pattern is not continuous where messages, in most cases, are short but frequent that requires sharing with other interested players within a time limit.

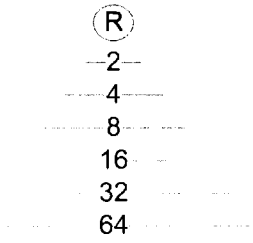

Figure 6.7: State information distribution through binary tree

Consider a system with N players where each player is a uniform distance away from each other. Say that each player, with its available bandwidth, can simultaneously relay messages to only two other players. The most commonly practiced approach is the binary tree to distribute state information. The root forwards the state to its two children, and each of the children in turn relays it to two other players and so on. According to Figure 6.7, it takes 6 time slots to complete the state-dissemination task. One of the limitations of such an approach is the unexploited bandwidth of the players in the following time slots, if they are unused. Consider Figure 6.7 again; at time slot 4, 8 players can forward state information to 16 other players. But there are 7 players (1+2+4) at the upper level who can perform the job. In this scenario, their slots are idle; off-course idleness depends on applications that are irregular. We can exploit those idle periods of the players even at the cost of a complex protocol. But when bandwidth is not abundant, this complex protocol could be valuable. The slot-by-slot state forwarding is shown in Figure 6.8. It means that in the first slot or

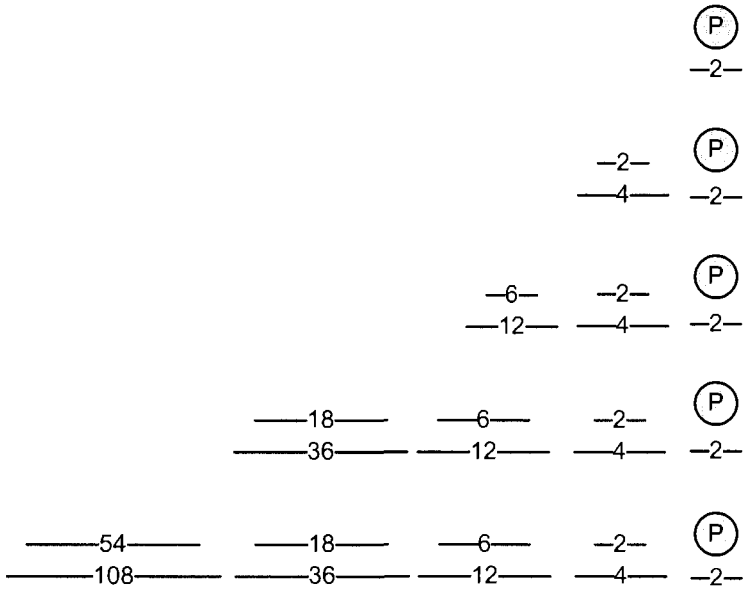

Figure 6.8: State distribution through expedited mechanism

round, the parent can forward the state information to two players. In the second slot or round, the parent and these two players who received the state information in the previous round can forward the state to six other players in total, and so on.

The key concept is to characterize a routing table that can share states among the players quickly. The core model of a player is shown in Figure 6.9. The objective is to disseminate state information among the players with a minimum number of time slots. Let all players be homogenous in terms of bandwidth and also equidistant from each other. This means that each player can be reachable to other players with the same time. Let each player have degree 1 (degree is an integer denoting how many players can receive state information from this player; it is a function of bandwidth). Thus the state-distribution process is like a linear approach. The source relays the state to player 0, player 0 forwards to player 1 and so on until player $N-1$ copies it to player N. Thus the state information floods the system within N time slots.

To improve such flooding, the following expedited approach is useful. Our task is to distribute the state information to N players. Say the source relays the state to player 0. Now the task can be considered two subtasks where the source and the player 0 each distribute the state to $N/2$ other players. So the pattern seems to be a recursive one, and this approach takes $log_2 N$ time slots rather than $O(N)$ time slots when each player has degree 1.

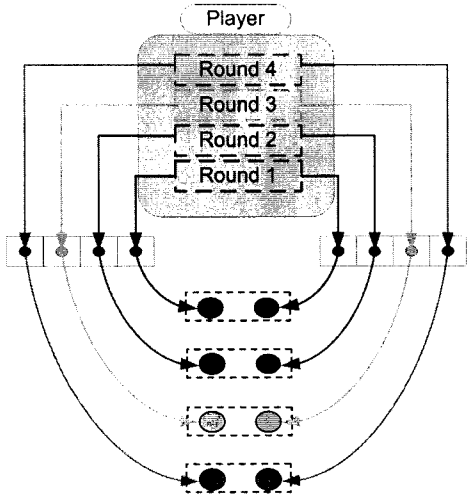

Figure 6.9: Core model of a player

6.2.1 Homogeneous Case

Let us move one step deeper, where each player has a degree of m. From a simple mathematical derivation, it is clear that a traditional tree-like approach will take $\lceil log_m N \rceil$ time slots. This could be good enough for many applications, but sometimes we need a quicker method. Let $p(i)$ denote the number of players have the state information at the beginning of the time slot i. Thus it takes the form $p(i) = (m+1) \times p(i-1) = (m+1)^{i-1}$. So the time required to distribute the message is $\lceil log_{m+1} N \rceil$. The difference between these two approaches is in the base of the logarithmic function.

6.2.2 Heterogeneous Case

It is complicated to model a system with heterogeneous aspects. In this case, we have to consider a case where players have different degrees. We can solve this through the same concept that we have used in the homogeneous case. Let us form groups whose members have the same degree. Say there are g groups of different degrees $m_1, m_2 \ldots m_g$ and each group has different players: $N_1, N_2 \ldots N_g$ players, respectively, where $N = \sum_j N_j$.

The order of state dissemination has an impact on performance. For better functioning, it is intuitive to forward state information to players with higher degrees. Let $m_1, m_2 \ldots m_g$ be ordered in descending order of degree. The time required to share state information with N_1 players of Group 1, each of whom has m_1 degree, is $\lceil log_{m_1+1} N \rceil$. Thus the next task is to distribute the state information to N_2 players of Group 2, each of whom has a degree of m_2 degree. At this stage of processing, there are N_1 sources. The time required to do each parallel task is $\lceil log_{m_2+1} N_2/N_1 \rceil$ by ignoring the extra degree $(m_1 - m_2)$ of each N_1 players. The time requirement would be $\lceil log_{m_3+1} N_3/(N_1 + N_2)) \rceil$ in the next step. So the total time to complete state dissemination would be:

$$t = \lceil log_{m_1+1} N \rceil + \lceil log_{m_2+1} N_2/N_1 \rceil + \ldots + \left\lceil log_{m_g+1} N_g / \sum_{j=1}^{g-1} N_j \right\rceil \qquad (6.2.1)$$

From this equation, it is clear that each subtask can be accomplished quickly even in the heterogeneous case. Of course, this requires special scheduling mechanisms to obtain the maximum benefit.

6.3 Quality Control

Quality control, a key requirement for online games, is a demanding task. Game providers put enormous resources into systems to guarantee the desired level of gaming experience. The client-server architecture is the dominant practice for online

games with a single authoritative server designed to handle game logic. Hence, if latency between a client and a server is large, the responsiveness of the game decreases, and the performance is likely to drop. In this section, I present a new quality-control algorithm considering players' virtual and geographical positions. The proposed procedure assumes that the interaction details between two players are inversely proportional to their virtual distance. Based on this principle and time constraint for the target application, a quality precedence matrix is formed to tune gaming experience. The intuition is to reduce latency by directly sharing application states between end-users rather than following a comparatively long path through the server, which is a variant of pure client-server architecture but incorporates players' participation actively.

6.3.1 Problem Formulation

Consider an MMOG system where a server (S) serves n players within a common area of interest. The application requires a kind of group communication within the common area of interest. In client-server architecture, each and every update message (game state) is relayed through the server. Let P be the set of players $P = \{p_1, p_2 \dots p_n\}$ and the end-to-end delay between a player p_i and the server S to be l_i. Let L be the set of end-to-end latency to the server for all players $L = \{l_1, l_2 \dots l_n\}$. Thus, the time required to forward a message from player p_i to another player, p_j, through the server is $L_{ij} = l_i + l_j$. Considering processing delay (D_p) at the server, it becomes $L_{ij} = l_i + l_j + D_p$. Thus, the pair-wise latency between any two players through a single-hop server can be presented by the matrix L_M.

The maximum latency can be determined by picking the two highest latencies to the server. Say l_{M_1} and l_{M_2} are the two highest latencies to the server for players P_{M_1} and? P_{M_2}. Thus the maximum latency will be $L_{max} = L_{M_1 M_2} = l_{M_1} + l_{M_2} + D_p$. The average latency can be defined by equation 6.3.1.

$$L_{avg} = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} l_{ij}}{\frac{n \times (n-1)}{2}} \tag{6.3.1}$$

Now, assume that T_c is the delay threshold constraint. Failure to obey the constraint deteriorates the performance of the application. Indeed, the performance remains steady as long as $L_{ij} < T_c$ for all i and j. Our goal here is to relax L_{max} for a better gaming experience using players' spare resources, i.e. minimize L_{max}.

6.3.2 Peer Potentials

One of the big problems of any collaborative group communication is the shortage of bandwidth. This problem gets even worse with peer heterogeneity. Thus, the proposed system requires confirmation of committed peer bandwidth. For an improved system, the players can directly exchange state information, in addition to sending it to the server. Thus, the system roughly estimates a number for each player, implying parallel message forwarding capabilities. We call this peer potentials or degree. If a peer has degree four, it could redirect messages to four peers.

6.3.3 Physical Position Approximation

To better comply with game specific time constraints, we need each player's physical position. An approximation concept can be incorporated to cover this assumption; in the process, each player will determines its distances to a few geographical landmarks placed strategically in the system. The distance to the landmark is going to be used for position estimation. It should be noted that these landmarks are not necessarily servers. It is intuitive that three landmarks would be adequate for such approximation. As the end-to-end delay itself is not invariant, the intersections may not be explicitly available. Thus, the physical location of a players' machine in the global coordinate system is roughly approximated. The server uses these estimates along with their degrees to comply with time constraints.

6.3.4 Virtual Position and its Impact

In online games, virtual and physical positions are two different concepts. Virtual position describes a location in a virtual world or in a game field. On the other hand, physical position means the actual position of a player on the Internet-for example, a player located in Berlin, Germany. Both are significant and need to be considered for an improved system:

- If two players are from two distinct geographical positions and are located far away from each other, their interaction in a virtual world could be challenging and timely state-sharing would be hard. In reality, physical position positively influences game performance;
- The virtual position of players in a game defines closeness. Area of interest is a well-known term in the gaming community. It forms a group for a set of closer players in terms of their virtual position. But we believe that even in the same area of interest, the importance of interaction for any two players is not uniform. Simply put, a closer player has greater importance than a distant one, even within the same area. For a better gaming experience, the importance of interaction must be taken into consideration.

`Assumption` - *In an area of interest, the importance (value) of interaction between two players is inversely proportional to their virtual distance, i.e.* $I_{i,j} \propto \frac{1}{d_{i,j}}$.

Thus a player must treat other players according to their virtual distance. A closer player would get more importance than a distant player. Considering this principle, a closeness (C) matrix is formed for a group of players.

6.3.5 Message Redirection Procedure

For an interaction space of n players, there are $C(n, 2)$ unique pairs for group communication. Thus, the matrix L_M has $C(n, 2)$ non-zero items representing the laten-

cies of all pairs. Let Pair-Wise-Delay (PWD) be a set that keeps such information. So,$PWD = \{L_{i,j} | \forall_{i,j} i < j\}$. This set assumes a symmetric latency relationship for any two players.

Say player p_i and p_j introduce the largest latency $L_{max} = L_{ij} = l_i + l_j + D_p$. The only way to reduce delay is to allow them to exchange messages directly. So, when L_{max} is at a critical stage, i.e. $L_{max} \geq T_c$, player p_i not only forwards update messages to the server but also to player p_j, and vice versa. The procedure can be applied to the next highest L_{max} as long as the degree of the players supports this. The main intuition is to allow the exchange of game states directly between two players that cause the largest latency in the system.

6.3.6 Quality Control

For a superior gaming experience, the latency and closeness matrices formed earlier can be used to identify where a quality improvement is significant. First, we normalize the latency matrix L_M by T_c, $N_L = L_M./T_c$. The closeness matrix is constructed from the virtual distances among players as explained earlier. Let $d_{i,j}$ represents the virtual distance between player p_i and player p_j. Say the virtual distance matrix is D and *maxDistance* defines the maximum virtual distance for a set of players. As a result, the closeness matrix becomes

$$C = \frac{(maxDistance + 1) - D}{maxDistance} \tag{6.3.2}$$

After normalization, the higher values of both matrices indicate the importance of quality control in terms of two different ratings. Now based on latency and closeness significance, wL and wC respectively, the quality precedence matrix $Q_p = wC \times C + wL \times N_L$ is formed where $wL + wC = 1$. Thus the game engine can tune Q_p according to its needs. The quality control algorithm is given in Figure 6.10. As the closeness matrix changes frequently, so does the quality precedence matrix. We suggest the periodic adjustment of the second part of the quality control algorithm.

Algorithm *Quality Control Algorithm*

Input

N: *Total number of players*

P: *Set of player*

D_p: *Packet processing delay at sever*

T_c: *Application-specific time constraint*

I_L: *Importance level*

begin

for each player $p_i \in P$ **do**

find geographical location, G_i, using landmarks

define end-to-end delay to server, l_i

define degree, d_i

define used-degree to 0, u_i=0

end for

form latency matrix L_m using $L_{ij} = l_i + l_j + D_p$

for each pair of players i and j and **do**

if ($L_{ij} > T_c$)

if ($d_i > u_i$ and $d_j > u_j$) **then**

$u_i = u_i + 1$

$u_j = u_j + 1$

adjust L_m using G_i and G_j (message redirection)

else 'relaxation is not possible'

end for

for each pair of players i and j **do**

if ($Q_{i,j} \leq I_L$ and $Q_{j,i} \leq I_L$) **then**

if ($d_i > u_i$ and $d_j > u_j$) **then**

$u_i = u_i + 1$

$u_j = u_j + 1$

adjust L_m using G_i and G_j (message redirection)

else 'relaxation is not possible'

end for

end

Figure 6.10: The quality control algorithm in MMOGs

The importance level (I_L) determines at what level of importance we should exchange direct messages. A lower value has a higher significance.

As mentioned earlier, we assume that the closeness of interaction between two players is inversely proportional to their virtual distance. Based on this principle and the time constraint for the target application, a quality precedence matrix is formed to improve the gaming experience. In adverse circumstances, this allows players to exchange state information directly. The idea is to reduce latency by directly sharing application states rather than following a comparatively long path through the server. The benefits are twofold: (1) the delay is reduced for some players, leading to a higher quality of experience; and (2) the load on the server is likely to drop, or it can make room for more players with the same resources.

Chapter 7

Evaluation

This chapter evaluates the hybrid MMOG architecture and identifies its pros and cons. It also covers the analysis of all new algorithms presented in the thesis such as multilevel multiphase load balancing, dynamic AoIM, expedited state dissemination, quality improvement and others. In this chapter, the efficiency and suitability of these ideas are verified for collaborative virtual applications, especially for online games.

As explained, the area of interest or a zone is a logical space containing a small part of the virtual world. Today, the size of a zone or an AoI is below 100 players (approximately 30—50). For example, *World of Warcraft* supports Raid-Groups [1] of up to 40 people. In our simulation, whenever the number of players is used as a variable in a figure, it refers to the size of an AoI; i.e., it describes the number of players in that zone. As can be seen, we have run the simulations for up to 500 players in a zone, a figure that is a big improvement over today's MMOGs, and can attest to the efficiency of our proposed protocol.

This chapter is organized into several sections. In Section 7.1, the effectiveness of intra-zone communication is given and compared with the well-known group communication strategy called NICE. The performance improvement, i.e. stability, due to

[1] http://www.worldofwarcraft.com/info/basics/raidarea.html

entity typing is illustrated in Section 7.2. The significance and robustness of interest-driven zone crossing is presented and discussed comprehensively in Section 7.3. The analysis and ranking of different load balancing algorithms are given in Section 7.4, while in Section 7.5, the feasibility and performance analysis of the dynamic area of interest management are presented. The importance of the expedited state dissemination is shown in Section 7.6. The quality improvement of an MMOG because of proper precedence ranking is illustrated in Section 7.7.

7.1 Effectiveness of Intra-Zone Communication

OPNET [2] modeler provides a modeling and simulation environment for designing communication protocols. It offers tools for creating and testing large network environments in software. Designer can test and optimize settings and determine network traffic values as required. The modeler comes with tutorials for the beginners with several introductory modeling projects. OPNET was used to model and simulate the architecture described in Section 4.1. We ran the software on *Intel Xeon* @ 3.00 GHz with 2GB of RAM that allows us to run simulations with up to 500 hosts. Here, 500 hosts represent the size of a zone in terms of players rather than the entire MMOG system. Indeed, depending on the model and value of some parameters, each host requires 3–6MB of memory. We simulate the behaviour of the intra-zone architecture of MM-VISA, and compare our proposed DS-ALM and NICE [BBK02] for 50 to 500 workstations in a zone. In selecting a well-known protocol for comparison, the following parameters are considered (based on the taxonomy in chapter 3): degree constraint, tree refinement, deployment level, P2P substrate requirement, overlay metric, mesh versus tree, and population. As can be seen from Table 7.1, NICE comes very close to DS-ALM in terms of such requirements.

NICE consists of layers where each layer is decomposed into clusters. Each cluster

[2] http://www.opnet.com/

Table 7.1: Protocol similarity

Points	NICE	MM-VISA
Degree constraint	Yes	Yes
Refinement	Periodically	Periodically (at join)
Deployment level	End-system	Hybrid
P2P substrate required	No	No
Exploit IP multicast	No	No
Metric	Delay and Bandwidth	Delay and Bandwidth
Mesh-First or Tree First	Mesh-First	Mesh-First
Population	100s of users	100s of users

may contain K to $3K-1$ members, and a leader is elected in each cluster. Each leader is allowed to access the immediate upper layer, and the root of the complete tree is finally the leader of the cluster in the highest layer. NICE has one parameter, K, the value of which has been set to 3 by the NICE designers. By contrast, DS-ALM has a parameter with which we can experiment: degree. In order to have a fair comparison, we compare NICE with a 3–7 degree version of DS-ALM. Indeed, most clients in NICE can only communicate with K to $3K-1$ (3 to 8) hosts. However, when NICE elects a leader, this leader will communicate with nK to $3nK-n$ members (where n is the number of layers to which the leader belongs). That means that for a tree containing four layers, and considering that the highest layer will only have one host, the root of the tree will have to send between 9 to 24 messages. Considering that most of the time the number of members within a cluster will be around $1.5K$ (since the clusters are split into two when they reach $3K$ members), the root will have to send more than 13 messages. Therefore, electing a bad leader can slow down the message-forwarding process. On the other hand, since DS-ALM allows its host to set the number of members to which it can forward messages, there will not be any risk

of slowing down the forwarding process.

7.1.1 Technical Considerations

Through OPNET, we deployed a random topology and constructed a tree structure of IP clouds with an average node degree between three and four. Leaves of the tree are subnetworks representing xDSL users. The tree is based on the real Internet structure, presenting different tiers (National, Regional, and Local). Considering this, all leaves are connected to local ISPs, i.e. tier 3. However, we made an exception for the rendezvous point and connected it to a tier 2 cloud in order to make it accessible faster. The different IP clouds are connected using PPP DS3 links (@45Mbps). The same kind of link is used between the routers and the clouds and between the routers and the xDSL modems. On the other hand, the end hosts (clients and server) are often connected to their modem/router through 100Mbps Ethernet cable.

All traffic generated by the clients is from the MMOG application only. We do not consider any background traffic because we want to compare raw results of the algorithms. Both applications only use UDP as the transport layer, which is quite logical since the purpose is to minimize the end-to-end delay while transmitting packets in real time. Moreover, the links in the simulation reflect real links since OPNET implements the correct behaviour of UDP/IP layers.

7.1.2 Real-world Game Traffic

To measure the performance of MM-VISA in a real gaming situation and to compare it with NICE, we have targeted simulations for two types of MMOGs: the Massively Multiplayer Online Role Playing Game (MMORPG) and the First Person Shooter (FPS). We have chosen to work with real-world measurements of traffic for these two types of games due to their varying characteristics. In this section, real-world traces are first introduced. Then, in the next section, we feed these traces into our

simulation to measure the performance of MM-VISA.

MMORPG and FPS games are somewhat similar as both require low bandwidth and generate small packets. The bandwidth requirement of an MMORPG is even lower because of its strategic nature and softer real-time constraints compared to an FPS. The game traffic of an MMORPG has strong periodicity but sometimes follows temporal locality.

The pattern of Client traffic is a complex function of different factors, but in most cases it follows exponential distribution like *ShenZhou*[3] online [CHHL05]. However, the packet inter-arrival time is approximately 600ms and varies across game genres. In *Quake*, the most computationally expensive part of a cycle is the rendering. It causes slower hosts to have significantly higher and more variable inter-arrival times, while the fastest hosts transmit most packets at 14ms intervals[LBA04][Bor00]. Moreover, the client's packet inter-arrival time also depends on the current map: some maps have very regular packet transmission intervals, and in others, packets are sent more randomly [LBA04]. Thus, the client traffic can be modeled as one extreme [Bor00] or two extreme distributions [CFSS05]. In our simulation, we considered one extreme distribution. The client traffic of *ShenZhou* and *Quake* are shown in Figure 7.1 and Figure 7.2, respectively, for an hour long simulation in two scales: w.r.t. seconds (top figure) and w.r.t. minutes (bottom figure). Observation reveals that on an average, each client generates more traffic in an FPS compared to in an MMORPG, as expected.

The aggregate traffic of a player, i.e. inbound traffic due to P2P nature, is shown in Figure 7.3 and Figure 7.4, for MMORPG and FPS respectively. The peak load in terms of inbound packets does not follow a linear shape. The variance of inbound packet for *ShenZhou* online and *Quake* are approximately 16.49 packet and 226.45 packet, assuming a player has 10 neighbours in the overlay. Note that the number 10 here is chosen simply to illustrate the traffic pattern. Thus, it is evident that an

[3]ShenZhou Online. http://www.ewsoft.com.tw/

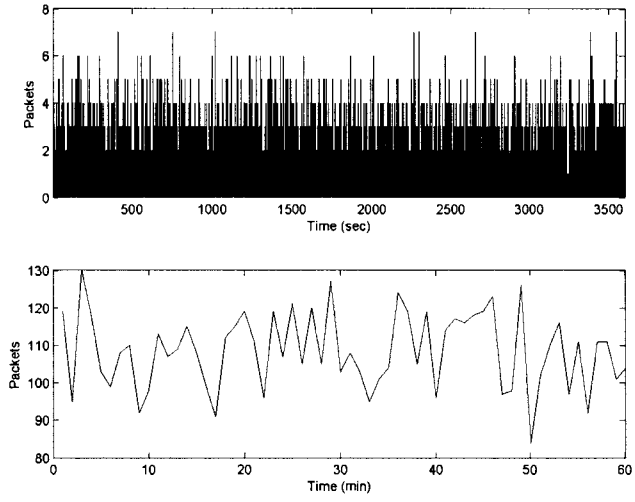

Figure 7.1: MMORPG client traffic from ShenZhou: exponential distribution, mean packet inter-arrival time = 550ms

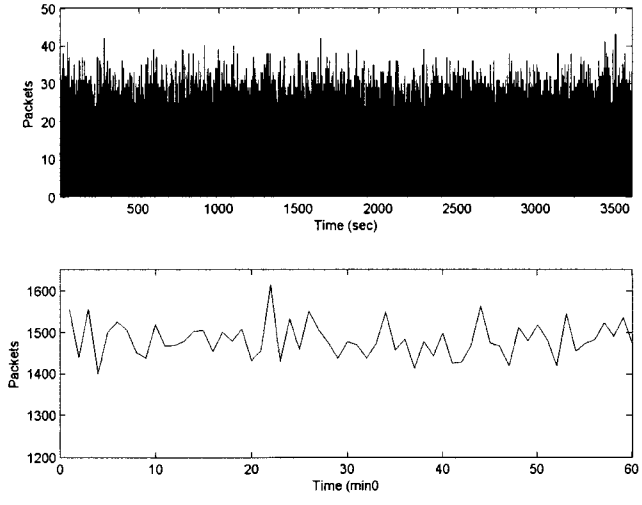

Figure 7.2: FPS client traffic from Quake: extreme distribution, mean packet inter-arrival time = 40ms

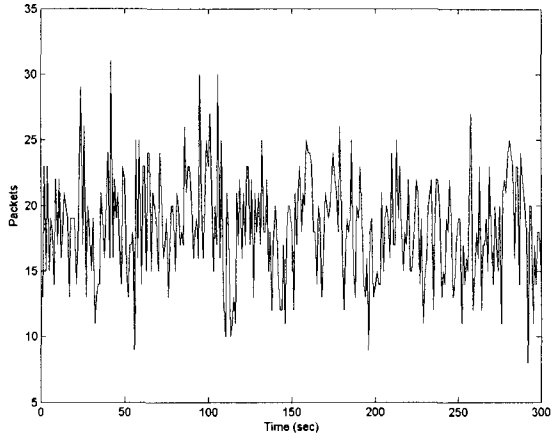

Figure 7.3: Inbound traffic for a hybrid MMORPG client with 10 neighbors (exponential distribution, mean packet inter-arrival time is 550ms for each neighbor)

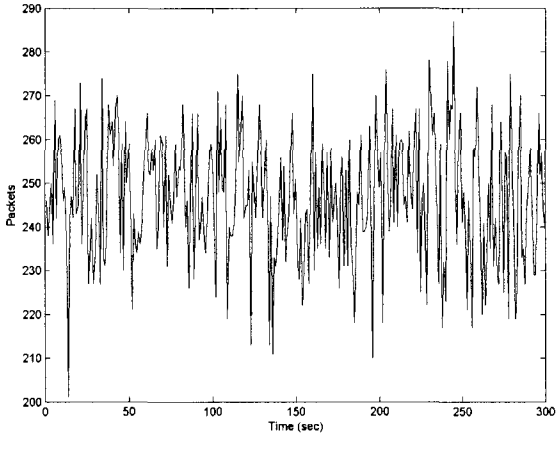

Figure 7.4: Inbound traffic for a hybrid FPS client with 10 neighbors (extreme distribution, mean packet inter-arrival time is 40ms for each neighbor)

FPS with hybrid MMOG architecture would be exposed to more loads compared to an MMORPG with the same architecture.

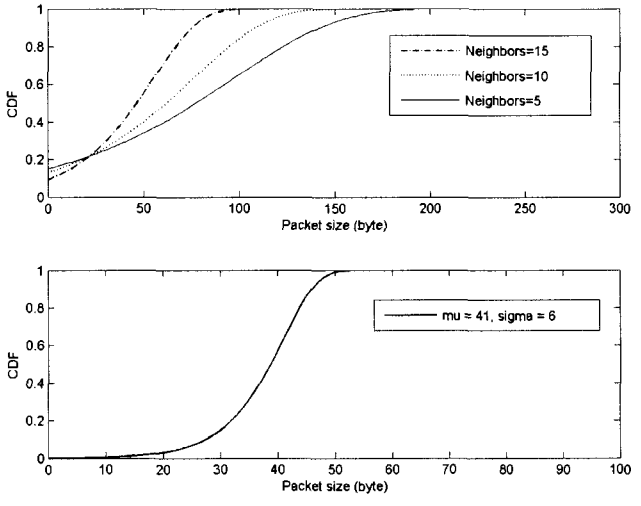

Figure 7.5: The cumulative distribution function (CDF) of the packet size of (a) a P2P player (b) one player of a client-server model

Due to the P2P component of this hybrid architecture, each player needs to relay some packets to some other players. Thus, the packet size of a player has a wide distribution and has a linear dependency on the number of direct neighbours. In *Counter-Strike*, the average packet size is 50.4 bytes and 6.15 bytes for each additional client without the UDP header [CFSS05][FCFW05]. The distribution has also been considered to have heavy tail behaviour. The extreme distribution with $a(n) = 34.5 + 4.2n$ and $b(n) = 9 + 3n$, where n is the number of players, is considered a good approximation for the server packet size. Exploiting that information to this model, the cumulative distribution function (CDF) of the payload is shown in Figure 7.5, the top part showing a player having ten neighbours and the bottom part for a single player. The client packet has an extremely narrow distribution with the mean size of

40 bytes [FCFW05]. Thus, an extreme distribution, with $a = 41$ and $b = 6$, can be a good choice for client packet size, leading to a packet size that is reasonable to carry the gaming functionality.

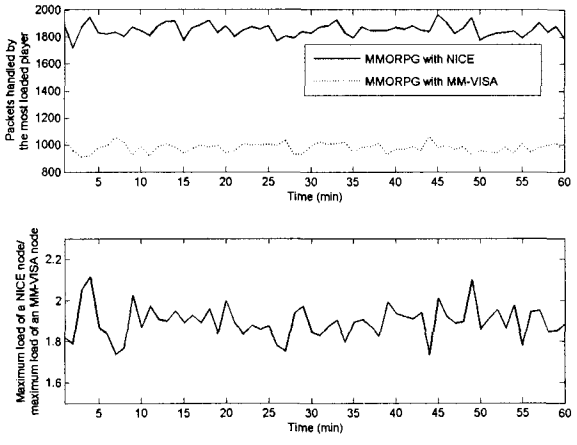

Figure 7.6: The worst-case load comparison between NICE and MM-VISA using MMORPG traffic

7.1.3 Simulations: DS-ALM vs. NICE

The above traffic models above were fed into the simulation system. A zone with 100 players was used for this experiment, where each player had an identical configuration and resource. After an hour-long simulation, we parsed the results and identified a node in the MM-VISA system that had the maximum load in terms of packets handled. A similar node was identified from the NICE configuration. The load of these nodes is compared and shown in Figures 7.6 and 7.7 (we randomly picked one minute from one hour to visualize the load variation) for the MMORPG and FPS, respectively. To present the difference, the MMORPG load is shown in minutes whereas the FPS load is shown in seconds. It is evident that the worst-case scenario

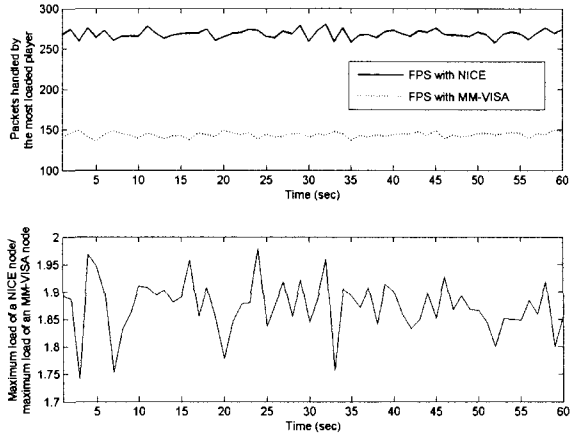

Figure 7.7: The worst-case load comparison between NICE and MM-VISA using FPS traffic

of an MMOG with NICE is more severe than that with MM-VISA. This is due to the layered feature of NICE where the cluster leaders in upper layers are more loaded for packet forwarding than the lower layers' leaders.

We compared the local communication strategy of MM-VISA, i.e. DS-ALM, against NICE. Both NICE and DS-ALM use different processes to exchange messages. This allows a certain degree of multitasking and therefore reduces the delays to forward packets. For NICE, we did not take into account that the leader of a cluster could change during the creation of the tree (of course, that does not prevent a new leader being created while splitting one cluster into two when the number of hosts exceeds $3K - 1$). However, this should not have a big impact on the results since the leader only sends messages to K to $3K - 1$ hosts, and all members of a cluster are the ones that minimized the end-to-end (ETE) delay between them.

The maximum-diameter diagrams, according to Figures 7.8 and 7.9, show that despite a higher number of hops used in DS-ALM, the end-to-end delay remains

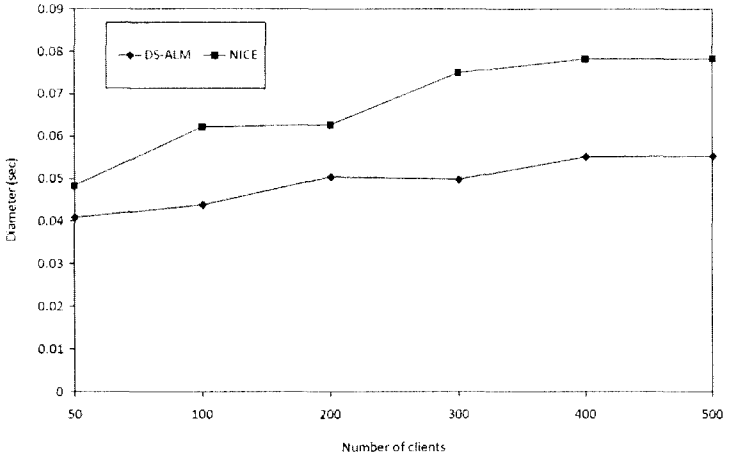

Figure 7.8: Maximum diameter in terms of delay

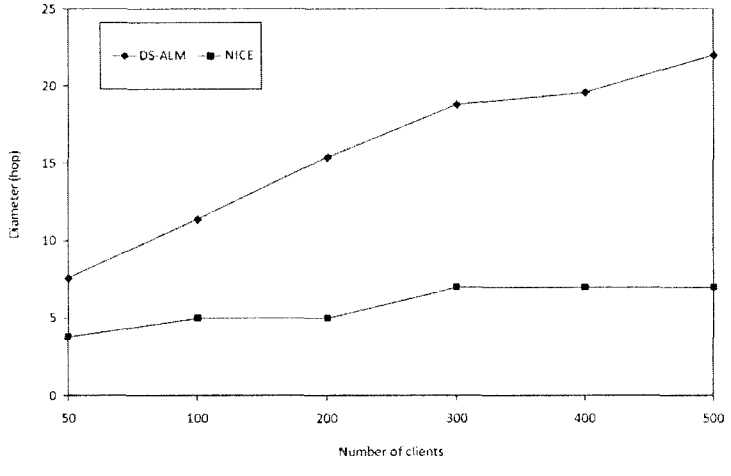

Figure 7.9: Maximum diameter in terms of hops

lower. This is possible because of the nature of each host. In DS-ALM, more nodes are considered gateway nodes while only the leader of a cluster is a gateway in NICE. In addition, since DS-ALM uses a greedy heuristic approach to define routing paths and creates ignore sets for each source, it ensures that, depending on the source, the gateway used to forward the message will be the one which minimizes the ETE delay.

Concerning the average end-to-end distance in terms of time (Figure 7.10), we can

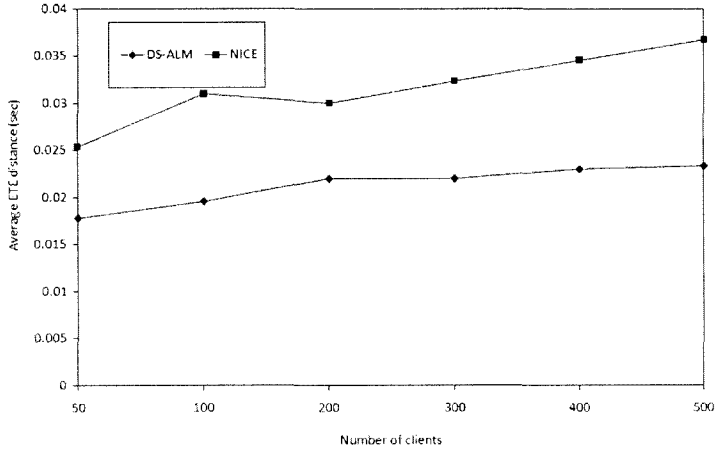

Figure 7.10: Average ETE distance in terms of time

observe that it is smaller with DS-ALM even if the average distance in terms of hops (Figure 7.11) is greater than that with NICE. However, the difference between the two algorithms is lower than that between the diameters and especially for the average number of hops. This difference concerning the number of hops can be explained by the fact that the diameter in terms of hops, in the case of DS-ALM, is an isolated value. The stretch is calculated by the formula $s = i/d$ (where i is the time taken to send a packet through a path established by the ALM algorithm and d is the time taken to send a packet in direct unicast). The results show that the average stretch is better in DS-ALM than in NICE (Figure 7.12). We can see that the two curves are

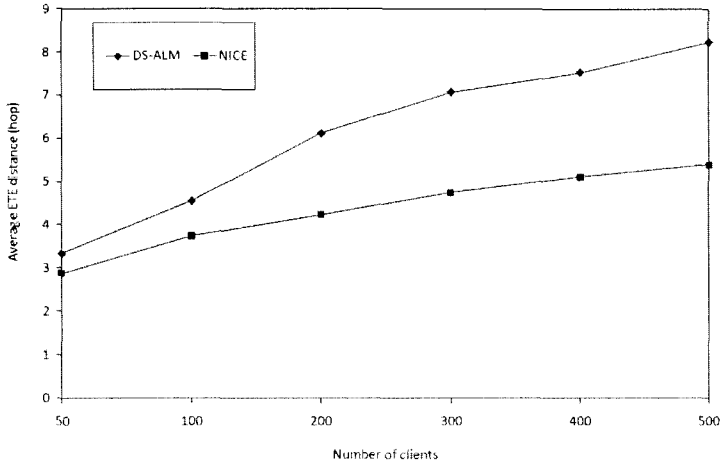

Figure 7.11: Average ETE distance in terms of hops

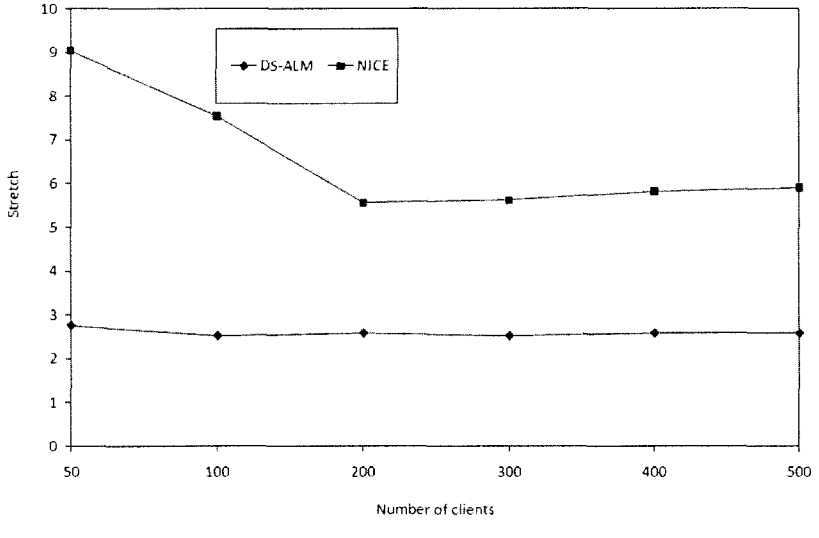

Figure 7.12: The comparison of stretch

parallel above 200 clients. We can suppose that for 50 to 100 clients, the real curves could be parallel too, but also that the simplification we made in the implementation results in larger stretch for NICE. However, the fact that the stretch is smaller in DS-ALM seems logical. Indeed, the average ETE is lower in DS-ALM, which will result in lower i values and therefore for the same d values; the stretch will be lower.

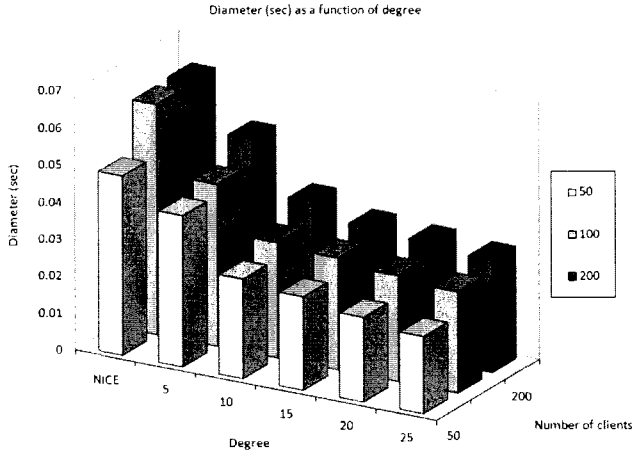

Figure 7.13: Comparison of diameter in terms of time (NICE & DS-ALM)

Figure 7.13 illustrates the comparison of NICE and DS-ALM for different degrees of parameter value. We can see the variation of the diameter in terms of time with the degree and the number of clients. This graph illustrates two concepts: the increase of diameter when the number of clients increases and the decrease of diameter for the same number of clients when the degree increases. We also observe that the diameter is lower for DS-ALM and can be as low as one half of NICE's diameter. The decrease in the diameter with the degree is easily understandable, as it is shown in Figure 7.14. When the degree increases, the number of connections a node can keep also increases. Therefore, some nodes will receive messages directly rather than indirectly and the end-to-end delay will decrease. It will also affect the number of hops, as we can see in Figure 7.14.

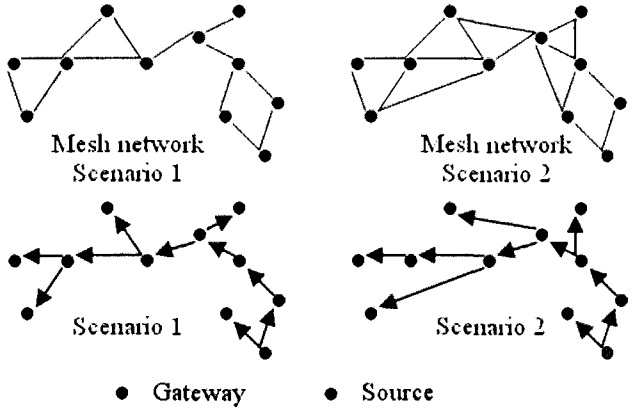

Figure 7.14: Effect of degree on the number of hops

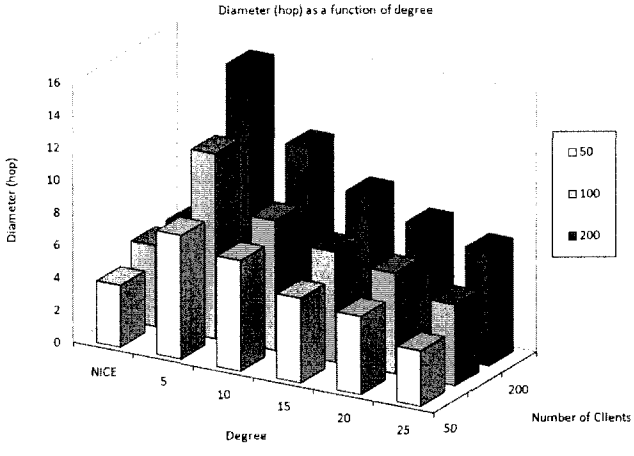

Figure 7.15: Comparison of diameter in terms of hop with the different degrees

Figure 7.15 illustrates the comparison between NICE and DS-ALM for different degree parameter values. We can see the variation of the diameter in terms of hops with the degree and the number of clients. As explained, the maximum number of hops decreases with the degree. Nevertheless, even with a degree of 25, NICE still has a smaller number of hops. This is due to its tree-based algorithm, while DS-ALM is meshed-based. Therefore, the tree constructed by DS-ALM will depend on the source and can result in a higher number of hops. However, as mentioned earlier, a minimum number of hops does not offer a minimum end-to-end delay diameter, as shown in the previous graphics.

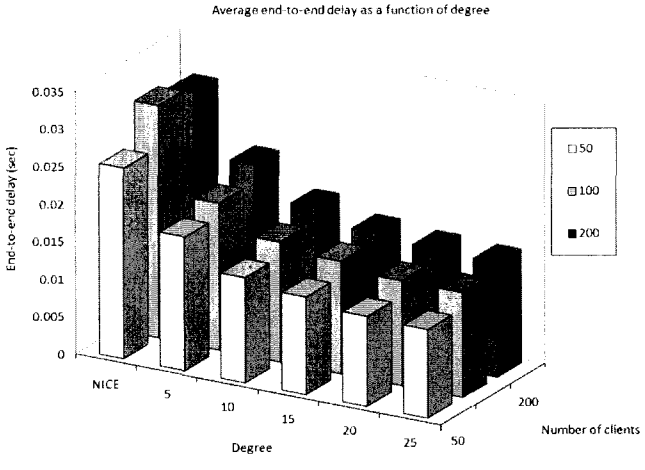

Figure 7.16: Average ETE evolution (NICE & DS-ALM)

Figure 7.16 illustrates the comparison between NICE and DS-ALM for different degree parameter values. We can see the variation of the average ETE in terms of time with the degree and the number of clients. We can make the same comments as we did with the diameter. The evolution of the average ETE with the degree is following the same curve, and the higher the degree, the higher the difference from NICE to go until DS-ALM shows an ETE that is half of that figured for NICE.

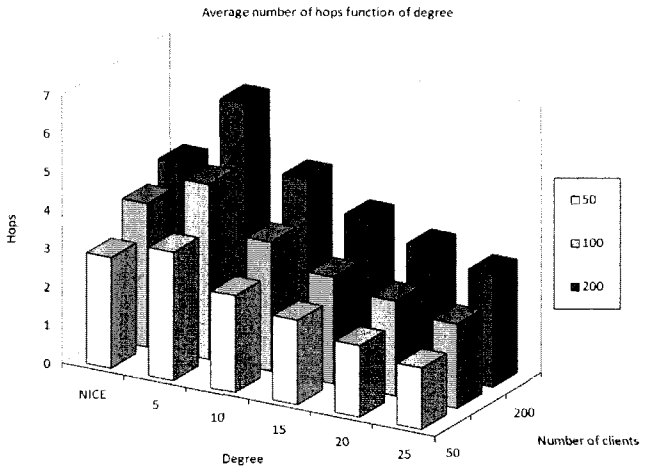

Figure 7.17: Average number of hops evolution with the degree (NICE & DS-ALM)

Figure 7.17 illustrates the comparison between NICE and DS-ALM for different degree parameter values. We can see the variation of the average ETE in terms of hops with the degree and the number of clients. In terms of average number of hops, the evolution is interesting. Indeed, when the degree is higher than 10, DS-ALM's average number of hops is below that of NICE. In addition, for a degree of 10, the difference is quite small. This is related to what we explained earlier concerning the highest isolated values obtained for the diameter. Besides this high diameter value, DS-ALM finds quite short paths, especially when the degree increases, since the number of possible paths increases considerably.

Like other parameters (diameter, number of hops, ETE), stretch decreases when degree increases (Figure 7.18). However, in term of ratio when compared against NICE (stretch of NICE vs. stretch of DS-ALM), the difference is quite large. The comparisons of show that DS-ALM is suitable for real time multi-source applications like MMOGs. Indeed, DS-ALM provides a lower diameter in terms of time as well as a lower average ETE delay. Even though the mesh is first established using the real

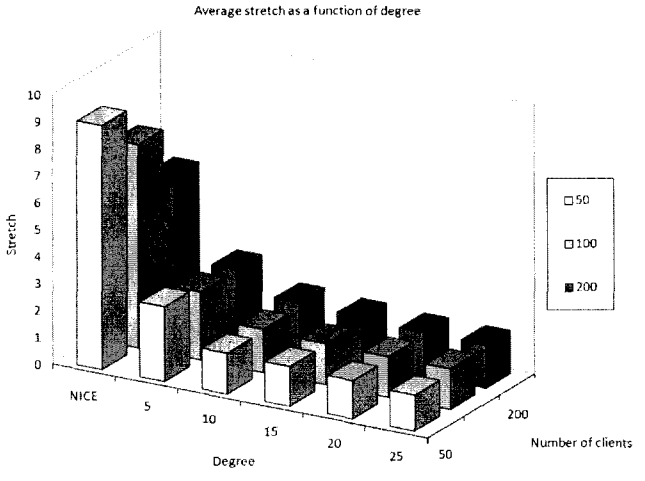

Figure 7.18: Evolution of stretch with the degree (NICE & DS-ALM)

geographic distances between nodes, the ALM paths are then based on the minimum ETE delay between the nodes depending on the source. DS-ALM also has better stretch values, which is a consequence of the lower ETE delay observed previously. This gets even better when considering a higher degree, the parameter of the DS-ALM algorithm, which lets each host to determine the number of clients to whom it can pass messages.

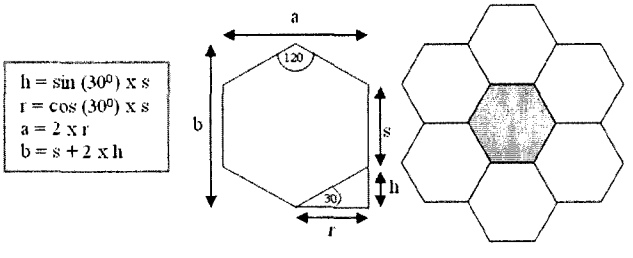

Figure 7.19: Simulation layout

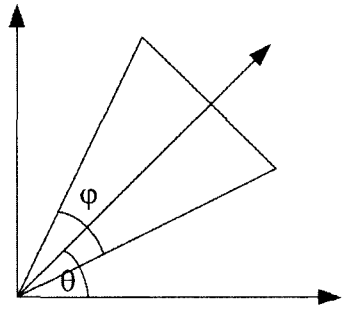

Figure 7.20: The avatar movement model

7.2 Stability Improvement

A simulation layout was designed to verify improvement due to entity typing, described in Sections 4.1 and 4.3. The performance and other related information are stated with respect to a hexagon, i.e. the center hexagon, which is surrounded by other six hexagons. The layout is given in Figure 7.19. Initially, players with difference types of physical characteristics are placed randomly in each zone. They can move from one zone to another zone. As mentioned earlier, we have tested the concept with respect to four different types of avatars. Each avatar is characterized by an average velocity with a movement pattern. The movement model is simply based on direction and coverage angle. The term "coverage angle" means the largest span at which an avatar can instantly change movement direction. Let a player's current direction, velocity, and coverage angle be θ, v, and φ, respectively. Thus, a player can instantly move within the span of $\theta - \varphi/2$ to $\theta + \varphi/2$. Figure 7.20 shows the object movement model used here. Some of the related simulation parameters are mentioned in Table 7.2. Four types of entities are considered: slow entities (e.g. soldiers on foot), normal entities (e.g. tanks), fast entities (e.g. jeeps) and very fast entities (e.g. jet planes). Each entity can be characterized by its velocity. The coverage angle is inversely related to velocity. Due to the law of inertia, the faster-moving entity

Table 7.2: The simulation parameters used for stability testing

Parameter	**Values**		
Hexagon length	1500 to 5000		
Object type	*Velocity (km/h)*	*Coverage(degree)*	*% of objects*
Slow	2	90	40%
Normal	5	45	30%
Fast	10	10	20%
Very Fast	50-350	5	10%

cannot make a sharp turn, but a soldier can. The simulation was carried out with discrete event simulation approach in the Windows XP platform. Each point of the following figures was the average of 20 to 25 runs. We present some relevant figures and analyze them in the following paragraphs. We have tested the advantage of entity typing for the zone layout presented in Figure 7.19. Performance improvement, i.e. the decreased number of affected nodes, during zone crossing is presented in Figure 7.21. Each point in the curve presents how many players are suffering with respect to the central zone in a given minute. It shows that for the same settings there are fewer affected players when we use the entity typing, i.e. clustering, concept. As entity type is used to classify the player and to place it into the appropriate cluster, it isolates different types of objects into different clusters and ultimately moves zone-crossing penalties from the whole zone to a small cluster. This improves the quality of the gaming experience.

The performance enhancement of this architecture is also tested against zone-crossing events. The total number of affected nodes in the general approach (A_g i.e. no clustering) and the clustering approach (A_c) are counted for an hour-long simulation. The improvement ratio A_g/A_c is shown in Figure 7.22. It reveals that

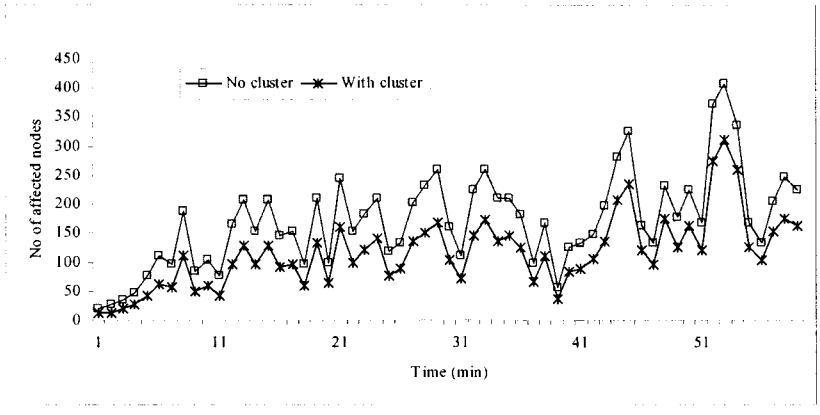

Figure 7.21: The analysis of entity typing concept

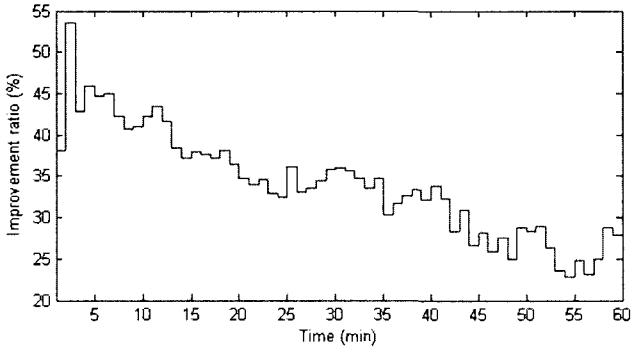

Figure 7.22: Improvement ratio with respect to no clustering mechanism

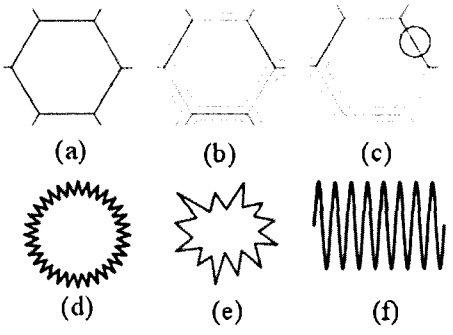

Figure 7.23: Various zone crossing models & movement patterns

Table 7.3: A comparison of crossing in different approaches

Model tested [Movement pattern]	**Number of crossings and relations**		
	Naive approach (C_n)	**VELVET** (C_v)	**MM-VISA** (C_m)
Any movement pattern	$C_n \geq C_v$	$C_m \leq C_v$	$C_m \leq C_n$
around the zone boundaries	$C_n \geq C_m$	$C_n \geq C_v$	$C_m \leq C_v$

there are fewer affected players in the clustering approach compared to the generic approach. But, the improvement ratio slowly drops because of comparatively lower Ag values. At the start of the simulation, an overlay is formed for a number of players in each zone. When players move from one zone to another, they break routing paths of their descendants. But, the moved players are usually placed at the edge of the new overlay. So, the value of the affected players drops over time hence the ratio.

7.3 Measurements of Interest-driven Zone-crossing

While it is difficult to define a simulation model to verify the interest-driven zone-crossing approach introduced in Section 4.5, we have tested the performance of dif-

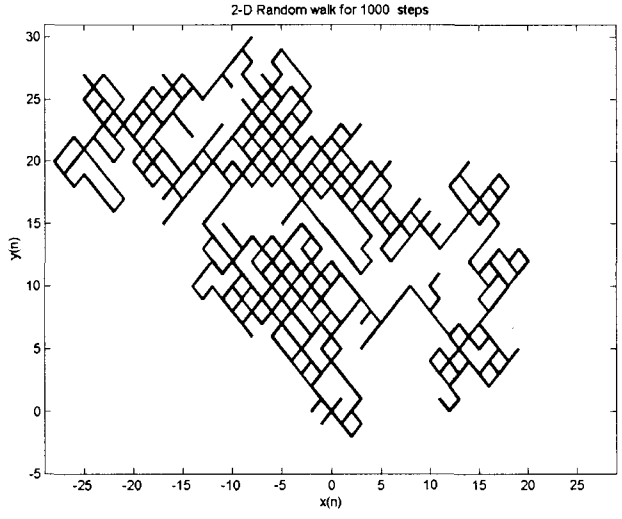

Figure 7.24: A random square-step walk

ferent types of zone-crossing approaches, such as simple (Figure 7.23a), overlapped (Figure 7.23b) and interest-driven, along with dynamic shared regions (Figure 7.23c) presented earlier. Obviously, performance depends on the size of overlapped regions and varies with movement pattern of the players like *32-point star* (Figure 7.23d), *explosion* (Figure 7.23e), *sinusoidal wave* (Figure 7.23f), random, etc. For all cases, the simple approach gives the worst result and the interest-driven approach is at least as good as the overlapped method because it is only applied when the overlapped method declares it a zone crossing. The improvement over the overlapped method is a function of a given interest vector and the position of players in the corresponding zones. The conditional performance improvement is given in Table 7.3.

We mentioned earlier that the performance of the presented zone-crossing approaches depends on the player's movement pattern and interest vector. We tested the presented methods for various movement patterns. Snapshots of two used random walks are given in Figure 7.24 and Figure 7.25 for 1000 and 12000 steps, respectively.

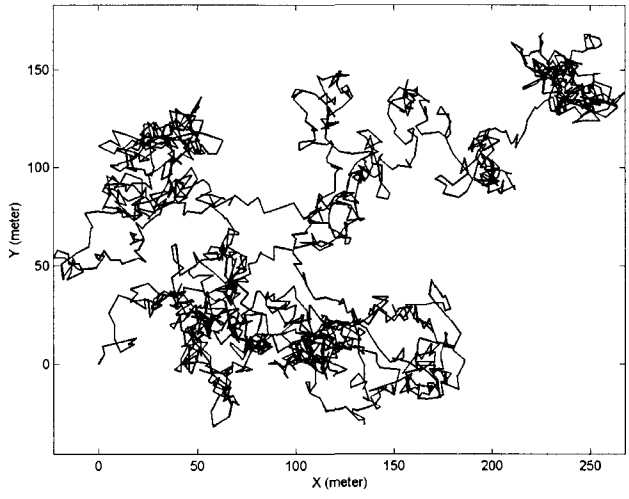

Figure 7.25: Another random walk

Many other walk-generating algorithms are used in the simulation.

We compared the performance of both VELVET and interest-driven zone-crossing methods for various buffer region sizes with identical setup and experimental data. In Figure 7.26, the X–axis presents the width of the buffer region while the Y–axis shows the number of zone crossings encountered. Each point of each curve is the average number of zone crossings of a single player for 1000 steps. The importance of the buffer region is clearly visible in Figure 7.26 as the number of zone crossings drops for the higher width of a buffer region. For the same settings, it also shows 95% *confidence interval* (CI) of both approaches. Moreover, the interest-driven zone-crossing method displays better performance than the exclusive buffer region approach. This confirms our earlier claim that interest-driven zone-crossing is at least as good as VELVET's approach or better. This is because of the delayed decision-making policy that depends on the interest vector of the concerned player and the density of players around.

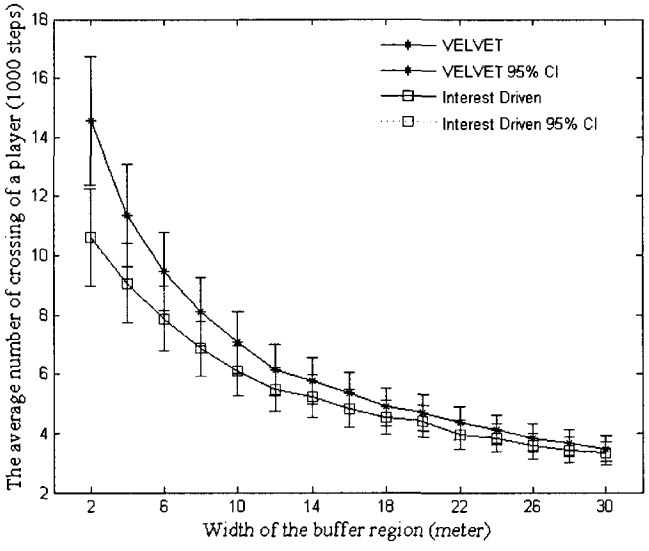

Figure 7.26: Comparison of zone crossing approaches for a player

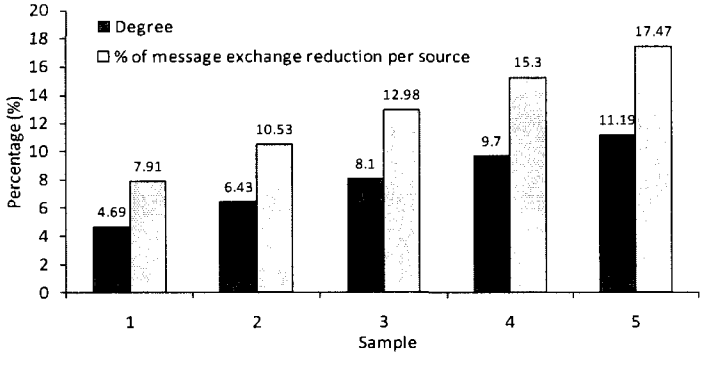

Figure 7.27: Message overhead reduction with respect to degrees

We also tested the message exchange reduction rate according to the section presented in Section 4.4. The improvement is closely aligned with degree, which is intuitive. As explained earlier, we can drop forwarding of regular messages when needed to the non-gateway nodes based on pruned links. Thus, if we have more non-gateway nodes, improvement will be higher. Figure 7.27 shows the message exchange reduction rate with respect to average degree. Thus for a moderate case, the improvement is around 10%.

7.4 Load-Balancing Performance

Extensive simulations were carried out to verify the load-balancing approaches proposed in Chapter 5 and to conduct the performance evaluation. We have targeted the simulations for two types of MMOGs: MMORPG (Massively Multiplayer Online Role-Playing Game) and FPS (First Person Shooter) and used the same traffic model explained earlier in Section 7.1.2.

We have defined load based on message generation rate. This definition is justified with the help of Figure 7.28 and Figure 7.29. Based on the population and packet models explained earlier, we have counted the number of players and the number of packets handled in each zone. Figure 7.28 shows players' distribution in 210 logical zones based on the parameters mentioned earlier. We randomly picked a zone, e.g. "27", and observed its adjacent zones. The number of players in that zone is shown using stairs. In Figure 7.29, we present the same scenario but in terms of packets. If we closely compare the lower curves of Figure 7.28 and Figure 7.29, it is apparent that the load of a server is not proportional to the number of players. Actually, it depends on different factors like game theme, type of actions involved in that zone, etc. Thus, flagging a zone as a hotspot based on the number of players is not entirely correct; it is ultimately a complex function of different attributes and varies across game genres.

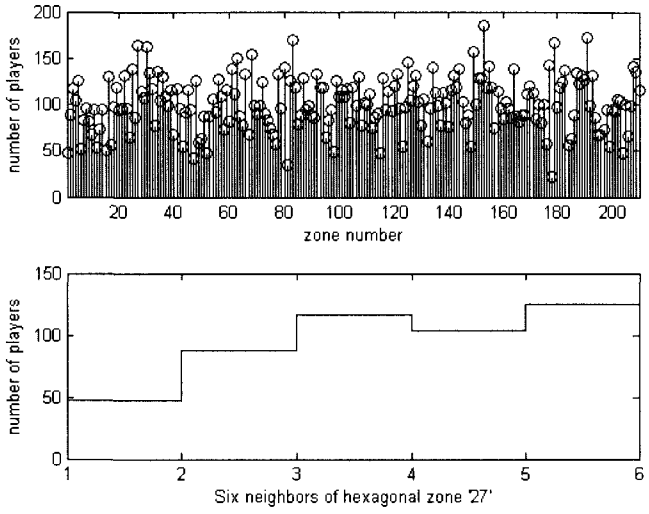

Figure 7.28: The load based on player counts

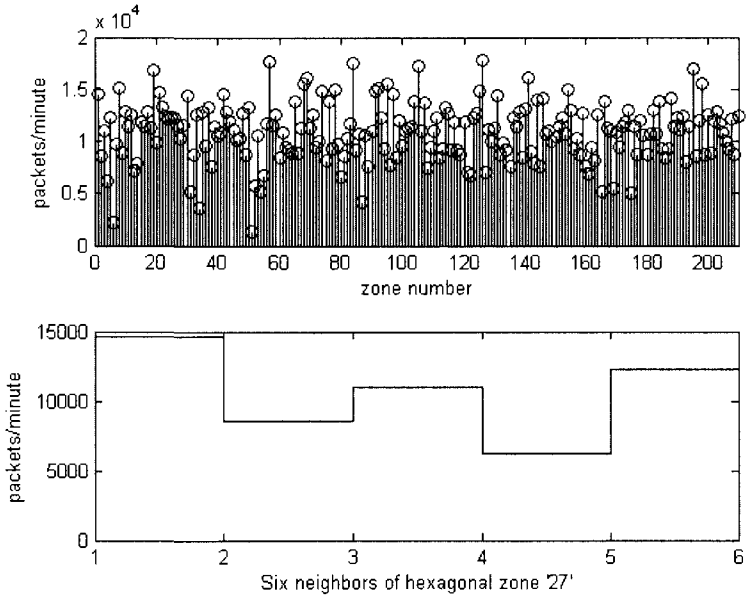

Figure 7.29: The load based on packet counts

Another interesting observation is related to the load of neighbors. If a zone is overloaded, it is likely that its surrounding zones are also loaded. Figure 7.30 displays the density of players in the entire game space (42 zones). A light zone is more heavily populated than a dark zone. It also shows there are two clusters of hotspots formed by three (3) adjacent loaded zones.

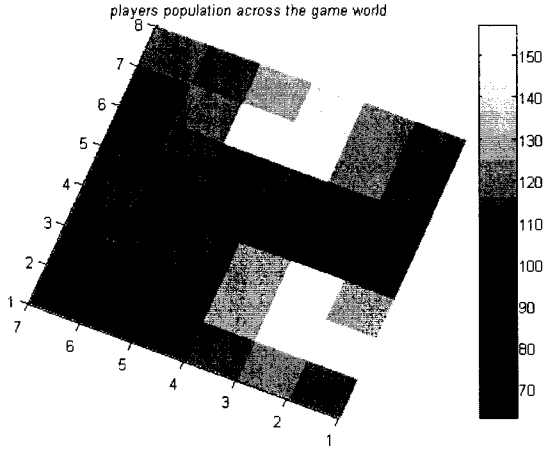

Figure 7.30: The density of players across zones

The effectiveness of the first phase of MMLB is shown in Figure 7.31 and Figure 7.32. For an hour-long simulation with identical setup, we have observed the system load at every minute. Figure 7.31 shows the standard deviation of packets being processed by all servers of an MMORPG system when there is no load-managing mechanism. It shows the uneven load distribution of the servers. The significance of the first phase of MMLB is the balanced load among the servers. From Figure 7.32, it is clear that the standard deviation of packets being handled by the servers has dropped to a large extent and is a good indication of the balanced load of the system.

We have also compared our algorithm with the generic zoning approach. We ran

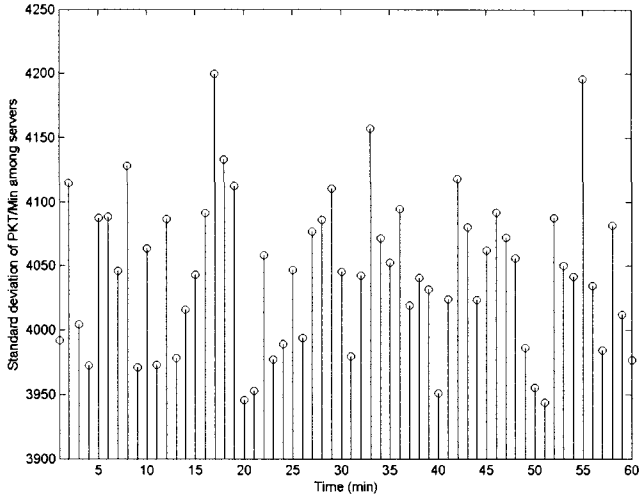

Figure 7.31: The standard deviation of the processed packets by the servers' overtime with no load-balancing mechanism

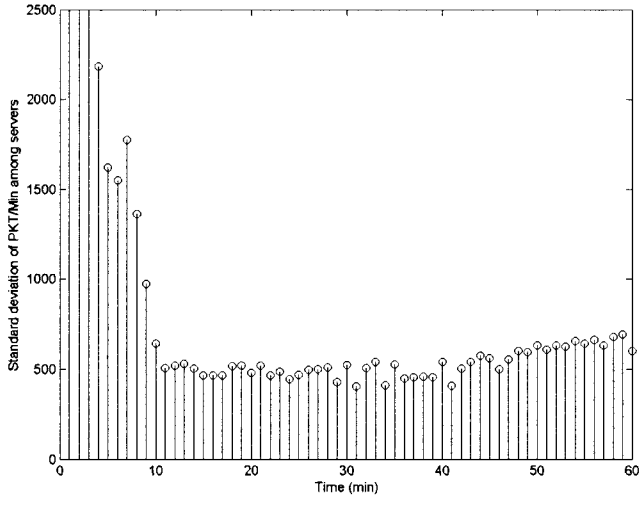

Figure 7.32: The standard deviation of processed packets by the servers' overtime using the first phase of MMLB

the simulation for 12 hours of game-play over 42 zones. Equations 5.2.2 and 5.2.3 are used to define T_m where p was set to 10. The generic load-balancing approach measures load as a function of the number of players in a zone while the MMLB uses number of packets processed to calculate load in a zone. We have shown already that the counting of processed packets is a better way to measure load. So, to be fair in the comparison, we tuned the generic approach to operate based on packets rather than player count. Figure 7.33 presents the number of hotspots encountered at every minute. The difference between the MMLB and the generic approach is clearly visible in terms of hotspots, as can be seen in Figure 7.33 top. We also plotted the ratio of hotspots (generic to MMLB) over time. The magnitude of the ratio is significant and justifies its superiority, as evident from Figure 7.33 bottom.

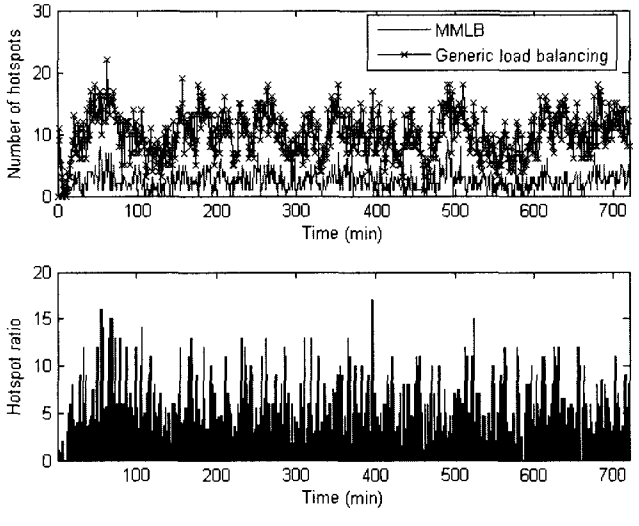

Figure 7.33: The evaluation of MMLB against the generic load balancing technique

The performance of MMLB depends on the magnitude of load reduced at each tuning phase. For 12 hours of game-play, we have counted the average number of hotspots encountered in every minute for a different scale of reduced load. According

to Figure 7.34, on an average there are 4 hotspots if MMLB releases 3% of the load of a hotspot zone. It continues to drop until 18%; is released however, interestingly, it gradually goes up again after that point, creating a U-shaped graph. This reveals that reducing load at any magnitude does not necessarily improve the gaming experience, as it adversely affects other zones.

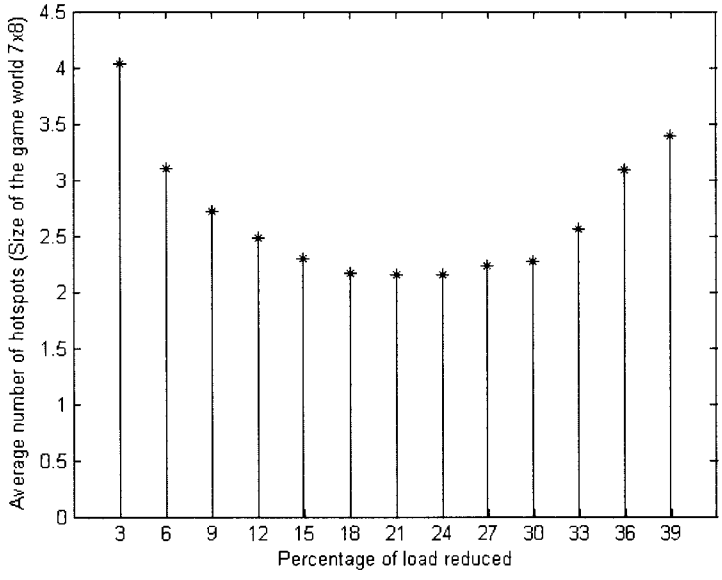

Figure 7.34: The evaluation of MMLB for different load handover magnitude

The presented adaptive zone shaping for load balancing is also evaluated. To have a fair comparison, we used identical settings in terms of zones, players and even packets. We then compared the adaptive technique against MMLB, which uses fixed hexagonal zones. The simulation was conducted for 12 hours (i.e. 12 hours of game-play). Figure 7.35 shows the number of hotspots encountered in every minute. We have plotted the scenarios from minute 600 to minute 720 to zoom in and clearly demonstrate the difference. For identical settings, it is interesting, and perhaps somewhat counter-intuitive, to note that the adaptive technique faces more hotspots than

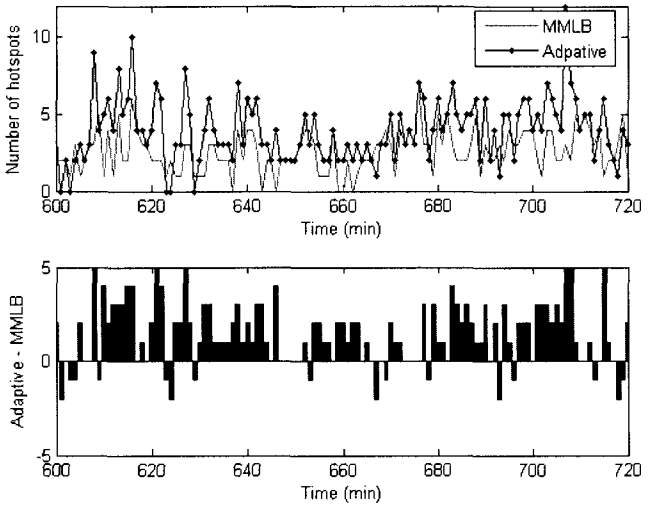

Figure 7.35: The comparison of MMLB and adaptive zone shaping with respect to hotspots

does MMLB. This is because of the restriction of cut movement. The difference is presented in Figure 7.35 bottom. It shows that there are a few points where the adaptive approach performs better, but overall MMLB performs much better against hotspots, especially in case of a series of hotspots, which is very important since, as mentioned before, the scenario of a series of hotpots is what happens in reality.

7.5 Evaluating Dynamic AoI Management

In this section, different points regarding dynamic area of interest management will be evaluated.

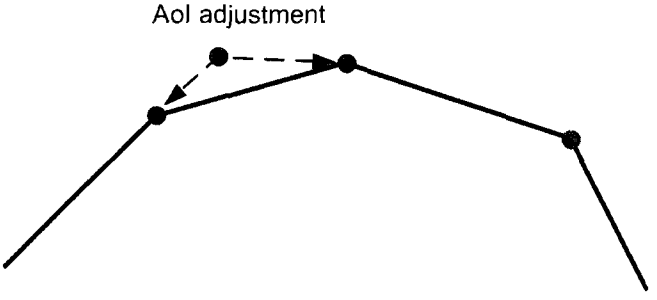

Figure 7.36: Two players need to be contacted to adjust AoI

7.5.1 Advantages of the Architecture

The dynamic area interest management technique presented in Section 6.1 incorporates peer-to-peer communications with the following advantages:

- It can reduce latency by directly sharing application data between participants rather than following a long path through a remote server;
- The traffic load on servers can be reduced significantly;
- The deployment and maintenance cost can be reduced due to distributed architecture;
- One-to-one features like instant messaging and voice communication could easily be included in the system, adding new features to the application.

7.5.2 Minimizing Communication Overhead and Improving Scalability

One feature of the proposed dynamic interest management method is the low system overhead. When the movements of a player cause a modification to an AoI (after satisfying two constraints, *safety-edge space* and *time-span*), the situation requires

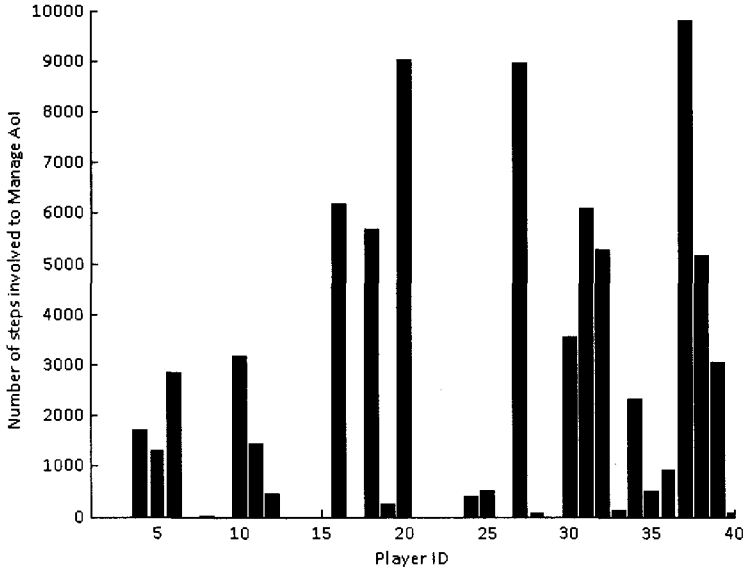

Figure 7.37: Distribution of AoI tracking responsibility in a group of 40 players

communication with only two adjacent players in the convex hull, as shown in Figure 7.36. These two players are the members of S_h responsible for interest management whose identities are known to all members of this AoI. As the architecture is peer-to-peer in nature, its cooperative design reduces communication and maintenance cost. This reduces load on the servers, which is required for scalability.

7.5.3 Dynamic AoI for Various Movement Patterns

Due to players' mobility, the performance of the interest management model is not static and largely depends on their movement patterns. We tested the presented methods for various movement patterns. A snapshot of two tested random walks was given in Figure 7.24 and Figure 7.25 for 1000 and 12000 steps, respectively. Eight other variant walk-generating algorithms were used in the simulation.

In traditional MMOGs, the group size is between 30–50, approximately. For

example, *World of Warcraft* supports Raid-Groups of up to 40 people. For a group of 40 players, we tested sharing of AoI tracking responsibility. The mobility model explained earlier was deployed for 10,000 steps for each player. The output is displayed in Figure 7.37, which shows that some players are more active than others maintaining the AoI. There are two reasons for this: (1) there is a moderate number of task hand-offs; and (2) the task is not distributed evenly. As the AoI is designed for peer-to-peer structure, so too many task hand-offs could be a problem, and thus uneven load distribution is actually a positive conclusion in this case.

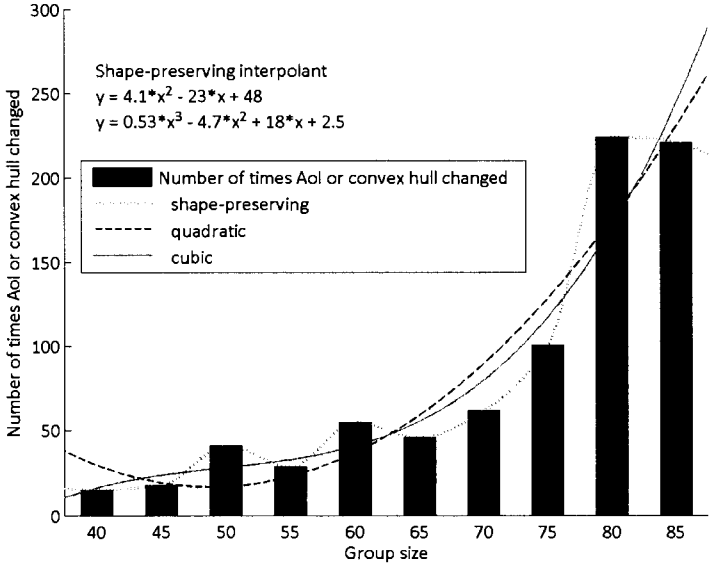

Figure 7.38: Performance of dynamic AoI for different group sizes - 10,000 movement steps

We also ran the simulation for different group sizes. The group size was changed from 40 to 85 with a step of 5. We measured how many times the convex hull was redefined for 10,000 movement steps, as presented in Figure 7.38. An interesting result for large groups was observed here: the values are incremented quite radically, which

implies that, for a large group, the convex hull changes quite frequently compared to a small group. The curve can be best fitted for quadratic and cubic functions, which are $y = 4.1x^2 - 23x + 48$ and $y = 0.53x^3 - 4.7x^2 + 18x + 2.5$, respectively.

7.6 Expedited State-Forwarding Improvements

In Section 6.2, a quicker state dissemination mechanism in the context of peer-to-peer MMOGs was presented. The theory was checked through the measurement of different timing parameters like the worst-case time required to share the game state among all players. In the following figures, we compare the expedited approach against the more commonly followed tree-like method. For measurement, both approaches (speedup and general) had identical settings in terms of degree, topology, node ordering (the higher bandwidth to the lower bandwidth), etc., unless otherwise mentioned.

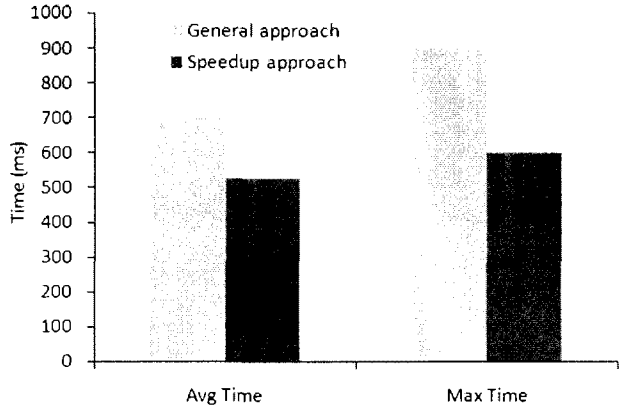

Figure 7.39: Comparing theories by measurement

In Figure 7.39, we have 512 players; each of them is 100ms away from each other (this scenario is unreal, its purpose being to verify the theory with the numerical values obtained through simulation) and has a degree of 2. The maximum time attribute

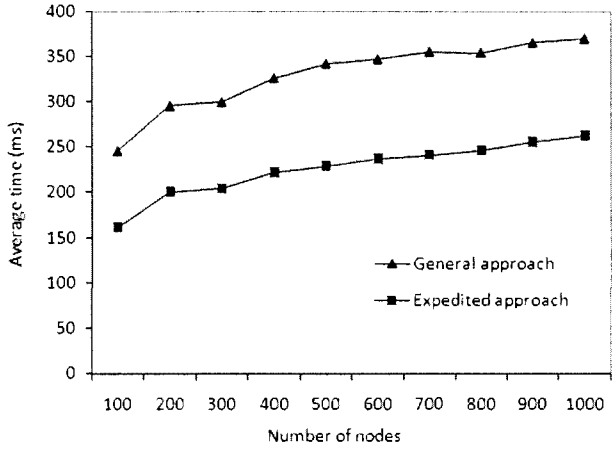

Figure 7.40: The measurement of average time to share state information

in Figure 7.39 clearly shows that the expedited approach accomplishes the task well ahead of the traditional approach. The average time found in Figure 7.39 also indicates that the expedited approach satisfies more players than general approach at a particular instant of time.

In Figure 7.40, we have tried to come close to real scenarios. The end-to-end delay between any two players is between 30ms and 100ms. The degree of each player is not identical: 20% of players have degree 4, 30% of players have degree 3 and the remaining 50% of players have degree 2. The key objective of such measurement is to check the significance of the proposed concept. Figure 7.40 votes for the expedited approach in terms of average time required for state dissemination for each different number of players. Figure 7.41 justifies the speedup approach in terms of maximum time required for state dissemination for different numbers of players.

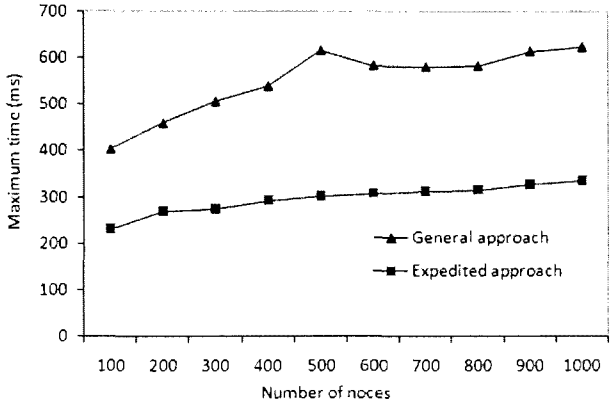

Figure 7.41: The measurement of maximum time required to share state information

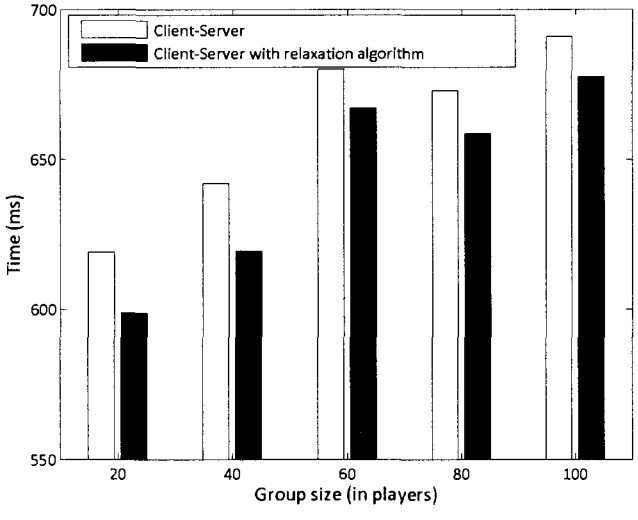

Figure 7.42: The drop-off of the maximum latency

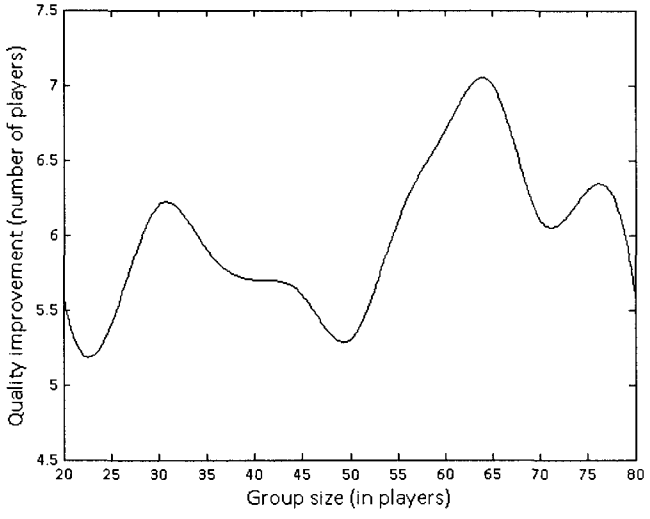

Figure 7.43: The maximum latency improvement for various group sizes

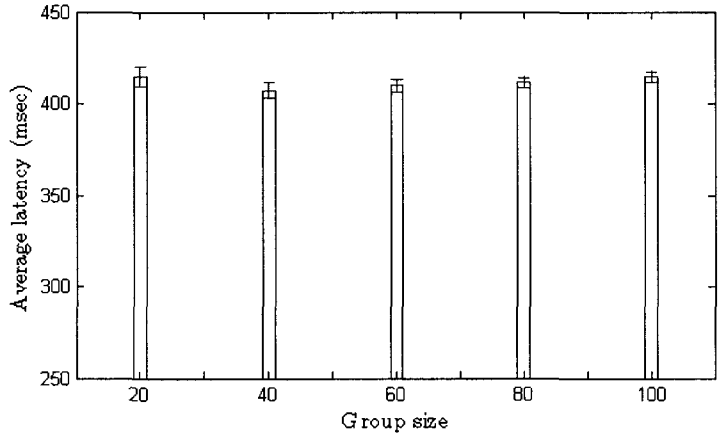

Figure 7.44: The average latency with 95% confidence interval for difference group sizes

7.7 MMOG Quality Improvement

We have verified the algorithmic improvement in terms of different timing parameters. For different group sizes (20–100 players), players were placed randomly and the server was approximated at the middle. For these settings, the presented quality control algorithm was applied to check its performance-the worst-case latency. The drop-off of maximum latency has been observed, as can be seen in Figure 7.42. Thus, it is clear that the time-constraint compliance procedure (first part of the algorithm) shortens time limit and players are likely to have a higher-quality gaming experience. On the other hand, we also checked how many additional players can comply with the timing constraint for various group sizes if the proposed concept is practiced. As performance and improvement is tightly coupled with players' distribution across the Internet, the real picture is hard to determine. But the improvement is apparent, according to Figure 7.43. For the same setup and different group sizes, the average latency and 95% confidence interval (CI) were calculated, as shown in Figure 7.44.

Chapter 8

Validation

In this chapter, we measure the performance of the system for different proposed algorithms and concepts, using prototype implementation and real-world networks. The hybrid MMOG architecture achieves scalability because of peers' participation. Unlike file-sharing applications, meeting a time constraint is a key requirement here. Thus, the length of a path in a tree cannot be too long; in fact, it has a limit. On the other hand, high dynamics of players can make a tree unstable. Maintaining a backup parent can improve the overall stability of the system. Challenges such as frequent movement at the zone boundaries and the hidden node problem were taken into consideration when implementing the system to ensure transparent users' experience. Many other points will be discussed in the following subsections.

8.1 End-to-end Delay Measurement

One simple scenario was set up where a node had to choose between two parents. The home internet users had cable Internet access (Rogers® ISP). These workstations calculated end-to-end delay 10 times. The average end-to-end delay for this case is shown in Table 8.1. Another workstation was set up in Dhaka, Bangladesh. This

Table 8.1: Reflecting the end-to-end delay on the ALM overlay network

Attribute	Case 1	Case 2
Average end-to-end Delay (ms)	20	28

station got a large end-to-end delay, 60ms. Sometimes some packets were even lost. This indicates the necessity of forming realms considering the geographic positions of the players.

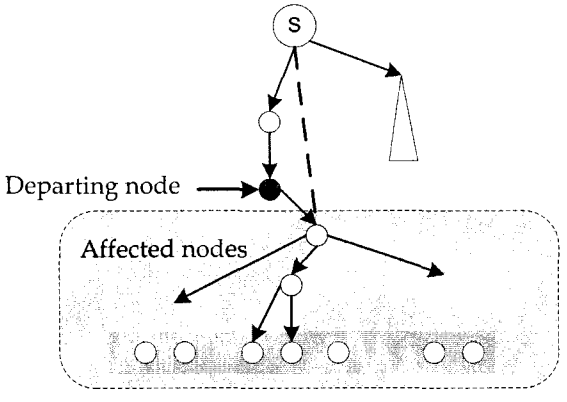

Figure 8.1: A scenario to measure overlay stabilization time

8.2 Overlay Stabilization Time Measurement

As mentioned earlier, introducing zones into the simulation map creates a problem of nodes having to disconnect from one overlay network and join another as they switch zones. This is called a graceful node departure, and it happens often in a simulation as players can move from one zone to another in the game world. Such a departure destabilizes the tree and causes it to reconstruct, with peers having to reconnect to resume exchanging state updates. Such reconstruction entails two events: the departing node has to join the new overlay network, and the orphaned

Table 8.2: The overlay stabilization time

Attribute	LAN	WAN
Overlay stabilization time (ms)	41	93

descendants have to rejoin their overlay network. A scenario was set up to measure the overlay stabilization process. This gives an idea of how bad it could be for a tree when nodes switch zone. In Figure 8.1, the black node is the departing node. The triangle represents a sub-tree of the original overlay connected to the master node. The descendent of the departing node has to reconnect to the tree. So, the descendent rejoins the tree as in a regular node-joining scenario. The interval between the node's departure and the time at which the descendent resumes receiving state update messages is recorded as Overlay Stabilization Time. The above scenario was run 10 times in a LAN environment, and 10 times in a WAN (Rogers® internet home users-same ISP) environment. The average times are shown in Table 8.2. However, our understanding is that the times could be higher when we have multiple IPSs involved. The results show that in a LAN environment, the Overlay Stabilization Time is 41ms, which can be easily tolerated in the simulation, taking 200ms as a threshold. In a WAN environment, however, the time is 93ms and can easily exceed the 200ms threshold. This negatively affects the performance of all of the descendants of the departing node.

8.3 Impact of Buffer Zone Size

We study how the size of the common area affects the number of disconnections and connections. Three random traversals were conducted at the boundary of two adjacent zones, as shown in Figure 8.2, where the black, white, and dotted curves represent random traversals. The height of the common area was then varied as a

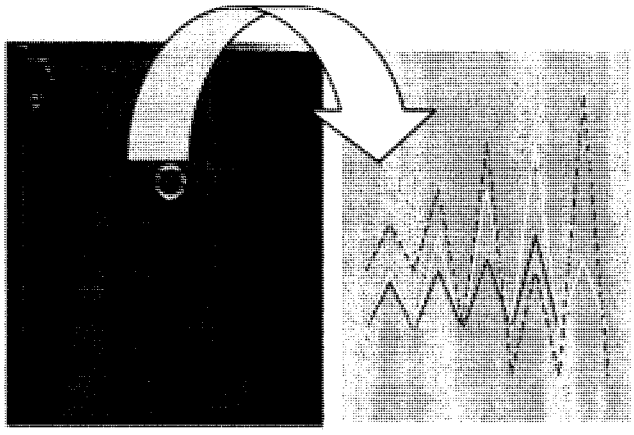

Figure 8.2: The common area between two zones with three random paths

percentage of the hexagonal zone's height. Figure 8.3 summarizes the results. The graph clearly demonstrates how the buffer zone significantly decreases the number of times the user has to disconnect from one zone and connect to another as it moves across the boundary of two adjacent zones. Also, the greater the height of the buffer zone (10% of the zone's height in our case), the fewer disconnections/connections happen.

8.4 Number of Tree-level

In case of players under the same ISP, a three-level tree can easily be supported and the expected delay from the master node to that third level could be about 120-140ms. A fourth level might also be supported with a slight violation of the 200ms threshold. But for players under different ISPs, the master would be able to support the third level with the maximum average delay around the threshold (200ms) or less. However, the fourth level would be difficult to support as players on the same ISP show a relatively small end-to-end delay. In reality, each player can support more

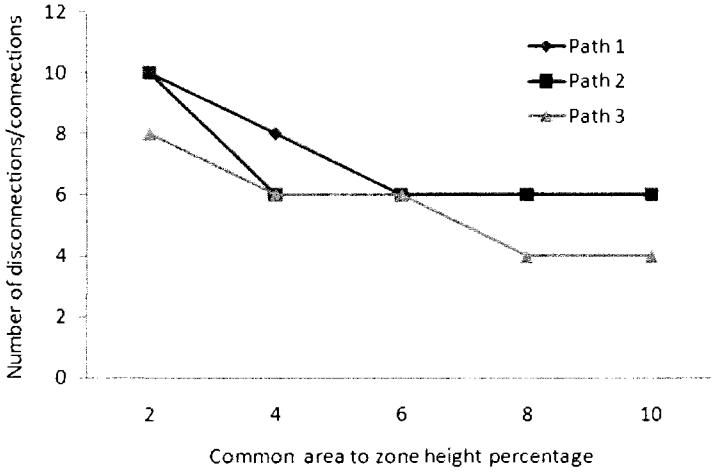

Figure 8.3: Performance improvement due to the use of common area

than one player, and thus the system expands exponentially. Thus, the incorporation of players in a system relaying game states has a clear advantage.

8.5 Validating Expedited State Forwarding Concept

To validate the expedited state-forwarding concept and to check peer-to-peer potential for MMOGs, we have planned to construct an overlay network over the Internet. The end-to-end delay was measured between two regions – the summary is given in Table 8.3. The table shows that there is a somewhat higher end-to-end delay between the Home PC (placed outside of the campus) and the Lab workstation (a workstation in 'Discover Lab'), even in the same region – Ottawa, Canada – than there is between the others. This is because of SSH tunneling, which was required to access Discover Lab's workstation outside the campus.

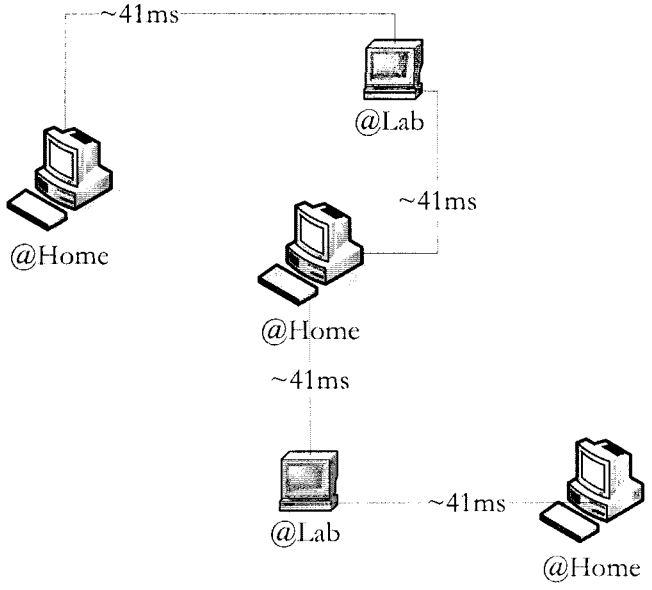

Figure 8.4: The path used for quality measurement due to expedited state forwarding

We ran multiple instances of the same program both at Home and Lab, and constructed a simple overlay path where each segment represents an overlay hop. The layout is given in Figure 8.4. Due to the P2P component of this hybrid architecture, each player needs to relay packets to others. Thus, the packet size of a given player has a wide distribution and has a linear dependency on the number of players. Here, the average packet size was 50 bytes and 6 bytes for each additional player without the protocol header.

The total time required to forward a game state through this P2P setup was 180-210ms. This reveals that we can build multiplayer online games over hybrid architecture, at least in considering players from a particular continent. According to the above experiment, it is evident that the performance can be improved using the *Expedited State-Forwarding* mechanism. But this performance depends on the number of hops in a path as well as their idle periods. In this setup, assuming that

Table 8.3: The measured end-to-end delay between regions

Region A	Region B	End-to-end delay (ms)		
		Minimum	Maximum	Mean
Home PC	Lab PC	41	45.5	41.5
Home PC	Dallas	27	31	29
Home PC	California	50	59	51.5
Home PC	Germany	68	79	73

each player gets a chance in every four packets to accelerate state forwarding, then the time required to forward a game state becomes 115.25ms. This clearly shows that the quality of the gaming experience can be enhanced, but that such enhancement depends on many factors like bandwidth, game type, etc.

Chapter 9

Conclusion and Future Directions

The term Peer-to-Peer (P2P) has become a common phrase referring to a combination of software applications, network technologies, and ethics of content sharing. In the years ahead, the evolution of the P2P concept will be spread across many applications. The networking industry will launch a wider range of peer-to-peer applications that should compete with traditional client/server systems for interest. The standard P2P protocol will undoubtedly gain wider approval. Finally, through the process of public debate, the consequences of free P2P applications (information sharing) on intellectual property law and copyright will slowly be resolved. There are several advantages to a P2P system, such as effective real-time collaboration and scalability. By contrast, a server cannot scale well with an increasing number of clients that require strong computing power and high-speed communications.

Distributed entertainment and interactive applications have become increasingly popular. The spread of broadband access among home users is also fueling the growth of online interactive applications. Besides file sharing, companies are also interested in content distribution, e-marketplaces, distributed search engines, distributing computing via P2P networks, and other applications. On the other hand, P2P nicely sets off social bonds known as social virtual worlds – worlds like *Second Life*, a digital

representation of the real world where human-controlled avatars evolve and interact in social activities.

9.1 Contributions

In this thesis, I presented a hybrid MMOG architecture, Massively Multiuser VIrtual Simulation Architecture (MM-VISA), that provides the properties of a centralized architecture, while exploiting P2P communication to achieve scalability. Different challenges of MMOGs in the context of peer-to-peer system are identified and their solutions are presented. The key objective of this virtual simulation model is to form a robust and scalable communication system for collaborative applications. The model integrates some of the benefits of centralized architecture and scalable distributed system. The P2P portion of the architecture has two main advantages. First, it has the potential to reduce latency by directly sharing game states between peers rather than following a comparatively long path through a remote server. Second, traffic load on servers can significantly be reduced. The latter will lead to a cost reduction in the deployment and maintenance of MMOGs.

The presented zone-based peer-to-peer multiuser simulation model and its associated algorithms can largely improve the quality of gaming. As the game world is divided into smaller manageable zones, over time a player usually moves from one zone to another zone, which requires the overlay network to be restructured. The restructuring changes the P2P overlay network causing routing problems. To control this problem, the proposed architecture uses players' gaming characteristics, such as their speed, to group them into clusters. As a result, a leaving player can only break routing paths within its own cluster keeping other clusters untouched, i.e. the routing problem is limited to a single cluster. Thus, player clustering can better stabilize the overlay network used in the zonal MMOGs. Many measures are taken for synchronous communication, such as intelligent interest-driven zone cross-

ing, state-sharing, etc. The thesis also gives several load-balancing methods for both uniform and non-uniform zonal MMOGs. The proposed load-balancing schemes identify a loaded server in terms of either packets processed per unit time or the number of players, and then move the load to other servers considering communication overhead and P2P overlay restructuring. The proposed multilevel multiphase load-balancing (MMLB) method is designed for fixed-size zones, works in multi-phases, and can be employed based on load-scale, in sequence or independently. MMLB reduces load in a step-by-step manner, and avoids problems associated with current load balancing schemes. We also present a novel load balancing scheme for non-uniform zones using a bisection procedure that does not adhere to any predefined zone size; i.e., zone sizes are flexible and can be determined dynamically.

An area of interest management technique for a peer-to-peer architecture is also outlined in this thesis. This technique assigns interest management duties to a subset of players for each AoI. The players who are inside the convex hull do not require regular interactions with the server because the peer-to-peer architecture is already managing the intra-zone communication. The presented concept can reduce communication load of the server not only due to the peer-to-peer nature but also due to the confined area of interest. Moreover, several effective solutions are given, including expedited state dissemination, opportunistic state forwarding to comply with time constraints, and others, in the context of improving the performance of peer-to-peer MMOGs.

9.2 Future Work

Application Layer Multicasting (ALM) has emerged as a serious alternative to IP multicast. While the transfer of multicast functions from routers to hosts addresses any problems associated with IP multicast, it also introduces problems in synchronization and stability. There is a need to analyze the stability problem in depth

considering the frequency of players leaving, the scope of players leaving and the time to reform the ALM tree. A stable ALM tree with low impact on node departure is an open research area that needs to be further explored.

An open issue for all ALM protocols is tree refinement: the reorganization or shuffling of the nodes in the tree. This is generally carried out to improve performance. In ALM, the quality of the path between any pair of members is comparable to the quality of the unicast path between that pair of members. Typically, a lower-diameter tree performs better than a higher-diameter tree. Hence, refinement is a way to improve the quality of an ALM structure once it is already constructed. The key point is that, if a node with zero out-degree joins a multicast session, the tree cannot be extended beyond that point which ultimately increases the height of the tree. An ALM protocol that is flexible to refinement has a clear advantage in this regard which is very much needed.

There are many challenges and unsolved problems in MMOGs. As players are given responsibility for relaying game states, there is greater opportunity to cheat. Since a cheater owns the machine on which he or she plays, such a player can easily tamper with the operating system and eventually the game, and can even evade anti-cheat algorithms. If cheating players succeed in gaining unfair advantages, the game quickly loses its appeal to other, honest players. This has a negative effect on subscription-oriented competitive commercial multiplayer games. This is an interesting and challenging research topic that needs to be further investigated and worked out with effective solutions.

It is important to devise a P2P framework that can survive even in a faulty network environment where peers dynamically join and disappear. It would be interesting constructing a multi-path communication framework for each pair of nodes considering network latency and processing delay. This is challenging especially in a gaming context, because games typically impose time-constraints on the communication layer that must be satisfied to ensure the quality of the applications.

In client-server architecture, the server pool performs the game operations and offers a consistent game world. In a hybrid P2P model, a player also depends on other players for the games states. So, consistency depends on both the serer pool architecture and the P2P system. The peer coordination and routing policy are important and its effective exploitation is challenging for the target application.

For cost-effective hybrid P2P applications like MMOGs, the appropriate incentive mechanism is a key element that promotes participants to share their resources. Game the ory tries to mathematically capture a strategic situation where one person's achievement in making choices depends upon the choices of others. This ideology can be applied for resource-provisioning purposes and for rationalizing output.

Bibliography

[Ale05] T. Alexander. *Massively Multiplayer Game Development 2 (Game Development).* Charles River Media, 2005.

[ASO07] D.T. Ahmed, S. Shirmohammadi, and J. Oliveira. Improving gaming experience in zonal mmogs. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 581–584, New York, NY, USA, 2007. ACM.

[AT06] M. Assiotis and V. Tzanov. A distributed architecture for mmorpg. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 4, New York, NY, USA, 2006. ACM.

[BBK02] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 205–217, New York, NY, USA, 2002. ACM.

[BKK+06] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Omni: an efficient overlay multicast infrastructure for real-time applications. *Comput. Netw.*, 50(6):826–841, 2006.

[BKV06] J. Boulanger, J. Kienzle, and C. Verbrugge. Comparing interest management algorithms for massively multiplayer games. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 6, New York, NY, USA, 2006. ACM.

[BLS07] E. Brosh, A. Levin, and Y. Shavitt. Approximation and heuristic algorithms for minimum-delay application-layer multicast trees. *IEEE/ACM Trans. Netw.*, 15(2):473–484, 2007.

[Bor00] M. Borella. Source models of network game traffic. *Computer Communications*, 23:403–410, 2000.

[BV95] F. Bauer and A. Varma. Degree-constrained multicasting in point-to-point networks. In *INFOCOM '95: Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies (Vol. 1)-Volume*, page 369, Washington, DC, USA, 1995. IEEE Computer Society.

[CC06] M. Claypool and K. Claypool. Latency and player actions in online games. *Commun. ACM*, 49(11):40–45, 2006.

[CDK+03] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, and A. Rowstron, A.and Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 298–313, New York, NY, USA, 2003. ACM.

[CDmKR02] M. Castro, P. Druschel, A m. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC*, 20:2002, 2002.

[CFSS05] C. Chambers, W-c. Feng, S. Sahu, and D. Saha. Measurement-based characterization of a collection of on-line games. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 1–1, Berkeley, CA, USA, 2005. USENIX Association.

[CGY00] S. Chen, O. Günlük, and B. Yener. The multicast packing problem. *IEEE/ACM Transactions on Networking*, 8(3):311–318, 2000.

[Cha00] Y.D. Chawathe. *Scattercast: an architecture for internet broadcast distribution as an infrastructure service.* PhD thesis, 2000. Chair-Brewer,, Eric A.

[CHHL05] K-T. Chen, P. Huang, C-Y. Huang, and C-L. Lei. Game traffic analysis: an mmorpg perspective. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 19–24, New York, NY, USA, 2005. ACM.

[CLR93] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms.* MIT Press, McGraw-Hill Book Company, 1993.

[CMB00] Y. Chawathe, S. Mccanne, and E.A. Brewer. Rmx: Reliable multicast for heterogeneous networks. In *In Proc. IEEE Infocom*, pages 795–804, 2000.

[CRZ02] Y-h. Chu, S.G. Rao, and H. Zhang. A case for end system multicast. *IEEE J. Sel. Areas Commun.*, 20(8):1456–1471, 2002.

[CWD+05] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, and C. Amza. Locality aware dynamic load management for massively multiplayer games. In *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 289–300, New York, NY, USA, 2005. ACM Press.

[DC90] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Trans. Comput. Syst.*, 8(2):85–110, 1990.

[DFC03] DFC intelligence. challenges and opportunities in the online game market. [online], 2003.

[DLL+00] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the ip multicast service and architecture. *Network, IEEE*, 14(1):78–88, 2000.

[Dou92] R.J. Douglas. Np-completeness and degree restricted spanning trees. *Discrete Math.*, 105(1-3):41–47, 1992.

[DSG+06] A. Diabi, S. Shirmohammadi, A. Gillmore, P. Lacombe, and J.C. de Oliveira. Internet-based collaborative virtual simulations with area of interest management. In *CTS '06: Proceedings of the International Symposium on Collaborative Technologies and Systems*, pages 200–207, Washington, DC, USA, 2006. IEEE Computer Society.

[DYNM06] N. Ducheneaut, N. Yee, E. Nickell, and R.J. Moore. "alone together?": exploring the social dynamics of massively multiplayer online games. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 407–416, New York, NY, USA, 2006. ACM.

[DZ03] T-N. Duong and S. Zhou. A dynamic load sharing algorithm for massively multiplayer online games. In *ICON '03: Proceedings of 11th IEEE International Conference on Networking*, pages 131–136, 2003.

[ERMS06] A. El Rhalibi, M. Merabti, and Y. Shen. Aoim in peer-to-peer multiplayer online games. In *ACE '06: Proceedings of the 2006 ACM SIGCHI*

international conference on Advances in computer entertainment technology, page 71, New York, NY, USA, 2006. ACM.

[ESRM03] A. El-Sayed, V. Roca, and L. Mathy. A survey of proposals for an alternative group communication service. *IEEE Network, special issue on Multicasting: an enableing technology*, 17(1):47–54, 2003.

[FCFW05] W-C. Feng, F. Chang, W-c. Feng, and J. Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Trans. Netw.*, 13(3):488–500, 2005.

[Fen97] W. Fenner. Internet group management protocol, version 2, 1997.

[FQL03] C. T. Fook, L. Qingping, and Z. Liang. A novel approach for addressing extensibility issue in collaborative virtual environment. In *CW '03: Proceedings of the 2003 International Conference on Cyberworlds*, pages 78–84, Washington, DC, USA, 2003. IEEE Computer Society.

[Fra99] P. Francis. Yoid: Extending the multicast internet architecture, 1999.

[GA04] M. Guo and M. Ammar. Scalable live video streaming to cooperative clients using time shifting and video patching, 2004.

[Gui08] E. Guizzo. The game-frame guild. *IEEE Spectrum*, 45(8):44–52, 2008.

[HASG07] M. Hosseini, D.T. Ahmed, S. Shirmohammadi, and N.D. Georganas. A survey of application-layer multicast protocols. *IEEE Commun. Surveys and Tutiruals*, 9(3):58–74, 2007.

[HBH06] T. Hampel, T. Bopp, and R. Hinn. A peer-to-peer architecture for massive multiplayer online games. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 48, New York, NY, USA, 2006. ACM.

[HCC06] S-Y. Hu, J-F. Chen, and T-H. Chen. VON: a scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, 2006.

[HDGM01] M. Bawa H. Deshpande and H. Garcia-Molina. Streaming live media over a peer-to-peer network, 2001.

[IEE98] IEEE. Ieee standard for distributed interactive simulation - application protocols, 1998.

[IHK04] T. Iimura, H. Hazeyama, and Y. Kadobayashi. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 116–120, New York, NY, USA, 2004. ACM.

[KAS07] I. Kazem, D.T. Ahmed, and S. Shirmohammadi. A visibility-driven approach to managing interest in distributed simulations with dynamic load balancing. In *DS-RT '07: Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 31–38, Washington, DC, USA, 2007. IEEE Computer Society.

[KC04] Y. Kim and K. Chon. Scalable and topologically-aware application-layer multicast. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1266–1270, 2004.

[KF02] M. Kwon and S. Fahmy. Topology-aware overlay networks for group communication. In *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 127–136, New York, NY, USA, 2002. ACM.

[Kla06] L. Klastrup. Death matters: understanding gameworld experiences. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, page 29, New York, NY, USA, 2006. ACM.

[KLXH04] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games, 2004.

[KR00] J. Konemann and R. Ravi. A matter of degree: improved approximation algorithms for degree-bounded minimum spanning trees. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 537–546, New York, NY, USA, 2000. ACM Press.

[KR03] J. Konemann and R. Ravi. Primal-dual meets local search: approximating mst's with nonuniform degree bounds. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 389–395, New York, NY, USA, 2003. ACM Press.

[LBA04] T. Lang, P. Branch, and G. Armitage. A synthetic traffic model for quake3. In *ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 233–238, New York, NY, USA, 2004. ACM.

[LLH02] D. Lee, M. Lim, and S. Han. ATLAS: a scalable network framework for distributed virtual environments. In *CVE '02: Proceedings of the 4th international conference on Collaborative virtual environments*, pages 47–54, New York, NY, USA, 2002. ACM Press.

[LLS+05] J. Liu, B. Li, H-R. Shao, W. Zhu, and Y-Q. Zhang. A proxy-assisted adaptation framework for object video multicasting. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(3):402 – 411, 2005.

[LPM06] F. Lu, S. Parkin, and G. Morgan. Load balancing for massively multiplayer online games. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 1, New York, NY, USA, 2006. ACM Press.

[LQF03] Z. Liang, L. Qingping, and C-T. Fook. Mobile agent-based architecture for large-scale cve. In *CW '03: Proceedings of the 2003 International Conference on Cyberworlds*, page 69, Washington, DC, USA, 2003. IEEE Computer Society.

[LTB04] E. Lety, T. Turletti, and F. Baccelli. SCORE: a scalable communication protocol for large-scale virtual environments. *IEEE/ACM Trans. Netw.*, 12(2):247–260, 2004.

[Man04] E. Mantion. Online gaming: Where the lost boys are. Technical report, 2004. In-Stat Research Report IN0401178IA.

[McF05] R. McFarlane. Network software architectures for real-time massively-multiplayer online games, Mcgill University, 2005.

[MLRS02] N. Malouch, Z. Liu, D. Rubenstein, and S. Sahu. A graph theoretic approach to bounding delay in proxy-assisted, end-system multicast. In *Proceedings of IWQoS*, pages 106–115, 2002.

[OG03] J.C. Oliveira and N.D. Georganas. VELVET: an adaptive hybrid architecture for very large virtual environments. *Presence: Teleoper. Virtual Environ.*, 12(6):555–580, 2003.

[PK99] K. Park and R. Kenyon. Effects of network characteristics on human performance in a collaborative virtual environment. In *VR '99: Proceedings of the IEEE Virtual Reality*, page 104, Washington, DC, USA, 1999. IEEE Computer Society.

[PSVW01] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. Almi: an application level multicast infrastructure. In *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, pages 5–5, Berkeley, CA, USA, 2001. USENIX Association.

[Pul99] J. Pullen. Reliable multicast network transport for distributed virtual simulation. In *DIS-RT '99: Proceedings of the 3rd International Workshop on Distributed Interactive Simulation and Real-Time Applications*, page 59, Washington, DC, USA, 1999. IEEE Computer Society.

[PWCS02] V.N. Padmanabhan, H.J. Wang, P.A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 177–186, New York, NY, USA, 2002. ACM.

[RES01] V. Roca and A. El-Sayed. A host-based multicast (hbm) solution for group communications. In *ICN '01: Proceedings of the First International Conference on Networking-Part 1*, pages 610–619, London, UK, 2001. Springer-Verlag.

[RMR+01] R. Ravi, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt Iii. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31:58–78, 2001.

[SA05] A. Steed and C. Angus. Supporting scalable peer to peer virtual environments using frontier sets. In *VR '05: Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, pages 27–34, Washington, DC, USA, 2005. IEEE Computer Society.

[SG01] S. Shirmohammadi and N.D. Georganas. An end-to-end communication architecture for collaborative virtual environments. *Comput. Netw.*, 35(2-3):351–367, 2001.

[SGMZ04] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 107–120, New York, NY, USA, 2004. ACM Press.

[SKH01] J. Smed, T. Kaukoranta, and H. Hakonen. Aspects of networking in multiplayer computer games. In *International Conference on Application and Development of Computer Games in the 21st Century*, pages 74–81, 2001.

[ST02] S. Shi and J. Turner. Routing in overlay multicast networks. In *IEEE Computer and Communications Societies (INFOCOM)*, pages 1200–1208, 2002.

[SYH04] A. Sobeih, W. Yurcik, and J.C. Hou. Vring: A case for building application-layer multicast rings (rather than trees). In *MASCOTS '04: Proceedings of the The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, pages 437–446, Washington, DC, USA, 2004. IEEE Computer Society.

[SZ99] S. Singhal and M. Zyda. *Networked virtual environments: design and implementation.* ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.

[TF05] V. Theoktisto and M. Fairn. Enhancing collaboration in virtual reality applications. *Computers & Graphics*, 29(5):704–718, 2005.

[THD04] D.A. Tran, K.A. Hua, and T.T. Do. A peer-to-peer architecture for media streaming. *IEEE Journal on Selected Areas in Communications*, 22(1):121–133, 2004.

[TMZ03] H. Trefftz, I. Marsic, and M. Zyda. Handling heterogeneity in networked virtual environments. *Presence: Teleoper. Virtual Environ.*, 12(1):37–51, 2003.

[TV04] A. Tumbde and S. Venugopalan. A voronoi partitioning approach to support massively multiplayer online games, The University of Wisconsin, CS 740 project, 2004.

[VBD07] M. Varvello, E. Biersack, and C. Diot. Dynamic clustering in delaunay-based p2p networked virtual environments. In *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 105–110, New York, NY, USA, 2007. ACM.

[VBV+05] B. Vleeschauwer, B. Bossche, T. Verdickt, F. Turck, B. Dhoedt, and P. Demeester. Dynamic microcell assignment for massively multiplayer online gaming. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–7, New York, NY, USA, 2005. ACM.

[WL99] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIALM '99: Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 7–14, New York, NY, USA, 1999. ACM Press.

[WSB03] A. Wierzbicki, R. Szczepaniak, and M. Buszka. Application layer multicast for efficient peer-to-peer applications. In *WIAPP '03: Proceedings of the The Third IEEE Workshop on Internet Applications*, page 126, Washington, DC, USA, 2003. IEEE Computer Society.

[XTBL03] Z. Xu, C. Tang, S. Banerjee, and S-J. Lee. Rita: receiver initiated just-in-time tree adaptation for rich media distribution. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 50–59, New York, NY, USA, 2003. ACM.

[YMYI05] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito. A distributed event delivery method with load balancing for mmorpg. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–8, New York, NY, USA, 2005. ACM.

[YV05] A. Yu and S.T. Vuong. MOPAR: a mobile peer-to-peer overlay rrchitecture for interest management of massively multiplayer online games. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 99–104, New York, NY, USA, 2005. ACM Press.

[ZH03] R. Zhang and Y. Hu. Borg: a hybrid protocol for scalable application-level multicast in peer-to-peer networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 172–179, New York, NY, USA, 2003. ACM.

[ZZKK01] S. Zhuang, A. Zhao, B.and Joseph, R. Katz, and J. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV '01: Proceedings of the 11th international*

workshop on Network and operating systems support for digital audio and video, pages 11–20, New York, NY, USA, 2001. ACM.