Solving Key Design Issues for Massively Multiplayer Online Games on Peer-to-Peer Architectures

Lu Fan

Submitted in fulfilment of the requirements
of the degree of Doctor of Philosophy
at Heriot-Watt University
in the School of Mathematical and Computer Sciences
May 2009

Abstract

Massively Multiplayer Online Games (MMOGs) are increasing in both popularity and scale on the Internet and are predominantly implemented by Client/Server architectures. While such a classical approach to distributed system design offers many benefits, it suffers from significant technical and commercial drawbacks, primarily reliability and scalability costs. This realisation has sparked recent research interest in adapting MMOGs to Peer-to-Peer (P2P) architectures.

This thesis identifies six key design issues to be addressed by P2P MMOGs, namely interest management, event dissemination, task sharing, state persistency, cheating mitigation, and incentive mechanisms. Design alternatives for each issue are systematically compared, and their interrelationships discussed. How well representative P2P MMOG architectures fulfil the design criteria is also evaluated. It is argued that although P2P MMOG architectures are developing rapidly, their support for task sharing and incentive mechanisms still need to be improved.

The design of a novel framework for P2P MMOGs, Mediator, is presented. It employs a self-organising super-peer network over a P2P overlay infrastructure, and addresses the six design issues in an integrated system. The Mediator framework is extensible, as it supports flexible policy plug-ins and can accommodate the introduction of new super-peer roles. Key components of this framework have been implemented and evaluated with a simulated P2P MMOG.

As the Mediator framework relies on super-peers for computational and administrative tasks, membership management is crucial, e.g. to allow the system to recover from super-peer failures. A new technology for this, namely Membership-Aware Multicast with Bushiness Optimisation (MAMBO), has been designed, implemented and evaluated. It reuses the communication structure of a tree-based application-level multicast to track group membership efficiently. Evaluation of a demonstration application shows

that MAMBO is able to quickly detect and handle peers joining and leaving. Compared to a conventional supervision architecture, MAMBO is more scalable, and yet incurs less communication overheads. Besides MMOGs, MAMBO is suitable for other P2P applications, such as collaborative computing and multimedia streaming.

This thesis also presents the design, implementation and evaluation of a novel task mapping infrastructure for heterogeneous P2P environments, Deadline-Driven Auctions (DDA). DDA is primarily designed to support NPC host allocation in P2P MMOGs, and specifically in the Mediator framework. However, it can also support the sharing of computational and interactive tasks with various deadlines in general P2P applications. Experimental and analytical results demonstrate that DDA efficiently allocates computing resources for large numbers of real-time NPC tasks in a simulated P2P MMOG with approximately 1000 players. Furthermore, DDA supports gaming interactivity by keeping the communication latency among NPC hosts and ordinary players low. It also supports flexible matchmaking policies, and can motivate application participants to contribute resources to the system.

*To my parents **Zhenyi Fan** and **Baozhu Yin**.*

Acknowledgements

Firstly, I am deeply indebted to my primary supervisor **Dr. Hamish Taylor** for his dedicated guidance, encouragement, inspiration and patience throughout my PhD. I heartily thank my second supervisor **Dr. Phil Trinder** for his strong and continuous support, invaluable advice and genuine enthusiasm for my work. Their direct assistance and encouragement has improved my work in many ways.

I gratefully acknowledge the contributions of **Jeff Hoye** for tirelessly providing vital software maintenance for the FreePastry, the Direct simulator, and for many helpful suggestions that expedited my research.

I would also like to acknowledge the Heriot-Watt University partial scholarship that made this research possible. Many thanks must go to the lab helper allocator **Jenny Coady**, admissions officer **Fiona Dick**, financial officer **Christine McKenzie**, computer officer **Iain McCrone**, and my colleague **Brian Palmer**, in the School of Mathematical & Computer Sciences at Heriot-Watt University for providing valuable assistance.

I take this opportunity to express my gratitude to my good friend **Nick Pilcher**. I owe much to him for helping me throughout the period of my research by providing support on my English language, proof-reading my publications, and giving me valuable suggestions to improve my writing skills.

Out with the work setting, I will be eternally grateful for the endless love and affection bestowed upon me from my family, my father **Zhenyi Fan** and my mother **Baozhu Yin**.

Finally, I am also grateful to many others, not all of whom can be named.

Declaration

I declare that this doctoral thesis was authored by myself and the work contained herein is my own, unless explicitly stated otherwise. The following articles were published during my research. Certain material and concepts from these publications will necessarily be presented within the body of this work.

- Lu Fan, Hamish Taylor and Phil Trinder, "Mediator: A Design Framework for P2P MMOGs". In Proceedings of 6th ACM SIGCOMM Workshop on Network and System Support for Games (Netgames '07), Melbourne, Australia, Sep. 19-20th, 2007, page 43-48. ACM Press, ISBN 978-0-9804460-0-5.

- Lu Fan, Phil Trinder and Hamish Taylor, "MAMBO: Membership-Aware Multicast with Bushiness Optimisation", 2nd International Conference on Distributed Event-Based Systems (DEBS '08), Rome, Italy, July 1-4th, 2008, short paper , pp 1-2.

- Lu Fan, Phil Trinder and Hamish Taylor, "Design Issues for Peer-to-Peer Massively Multiplayer Online Games", 2nd International Workshop on Massively Multiuser Virtual Environments (MMVE), at IEEE Virtual Reality (VR '09), Lafayette, Louisiana, March 15, 2009, pp. 1-11, to be published in International Journal of Advanced Media and Communication, ISSN (Online): 1741-8003.

- Lu Fan, Phil Trinder and Hamish Taylor, "Deadline-Driven Auctions for NPC Host Allocation in P2P MMOGs", 2nd International Workshop on Massively Multiuser Virtual Environments (MMVE), at IEEE Virtual Reality (VR '09), Lafayette, Louisiana, March 15, 2009, pp. 1-7, to be published in International Journal of Advanced Media and Communication, ISSN (Online): 1741-8003.

Contents

List of Figures

List of Tables

Glossary

Notation	Description	
ALM	Application-Level Multicast	55, 65, 74, 119, 175
AOI	Area of Interest	47, 83, 176
BM	Boot Mediator	88
BO	Bushiness Optimisation	120
COR	Communication Overhead Reduction	130
CVE	Collaborative Virtual Environment	16
DDA	Deadline-Driven Auctions	84, 146
DHT	Distributed Hash Table	61
DIS	Distributed Interactive Simulation	19
DR	Dead Reckoning	19
DVE	Distributed Virtual Environment	16
EDF	Earliest Deadline First	162
HC	Heterogeneous Computing	151
HLA	High-level Architecture	19
IAA	“I am alive” message	126
IMM	Interest-Management Mediator	83, 91
IM	Interest-Management	46
IVE	Immersive Virtual Environment	16
JAd	Job Advertisement	97, 158

Notation	Description	
MAMBO	Membership-Aware Multicast with Bushiness Optimisation	13, 55, 82, 84, 110
MAM	Membership-Aware Multicast	120
MFDQ	Multilevel Feedback Delay Queue	163
MMOFPS	Massively Multiplayer Online First-Person Shooter	16
MMORPG	Massively Multiplayer Online Role-Playing Game	16
MMORTS	Massively Multiplayer Online Role-Playing Game	16
MUVE	Multi-User Virtual Environment	16
NPC	Non-Player Character	11, 20, 59, 146
NVE	Networked Virtual Environment	16
PC	Player Character	20
POMP	Peer-to-Peer Online Market Place	120
PW	Persistent World	17
PvN	Player-vs.-Non-Player	20
PvP	Player-vs.-Player	20
QoS	Quality of Service	20, 24, 157
RAd	Resource Advertisement	94, 158
RM	Resource Mediator	84, 91
RQ	Resource Queue	161

Notation	Description	
TC	Trusted Computing	194
TQ	Task Request Queue	161
VE	Virtual Environment	12, 16
WoW	World of Warcraft	1, 7, 16, 20
ZM	Zone Mediator	89

— Well begun is half done.

Hesiod

Chapter 1

Introduction

1.1 Context

1.1.1 Massively Multiplayer Online Games

Computer and video games form a unique industry and research field that require non-trivial creativity and imagination. Within game studies, there is a lack of consensus on accepted formal definitions for game genres, so any individual game genre classification struggles to be accepted as complete or comprehensive. Among numerous game categories, this thesis is especially interested in what are widely referred to as "Massively Multiplayer Online Games (MMOGs)" due to their distinct characteristics, unique complexities, and significant commercial, social and academic impacts.

The history of MMOGs spans over thirty years, and the origins of its antecedents, Multi-User Dungeons (MUDs), can be traced back even further to the 1970s. A MUD is a text-based adventure game that implements a fantasy world as a collection of "rooms". Usually, a MUD provides a simple user interface that prompts its player with a descrip-

Figure 1.1: MMOG - the past (British Legends) and the present (World of Warcraft)

tion of the current room that the player is in, as well as objects, other players, or non-player characters in the vicinity. Actions, such as slaying monsters, completing quests, moving among rooms, and talking with other players, are performed by typing corresponding commands. Modern MMOGs have made a revolutionary improvement to this early and awkward format by introducing 3D graphical representations of virtual worlds and player avatars (Section 2.2.2). Modern MMOGs can support many more players who inhabit, communicate, cooperate and compete with each other on a large scale in a shared world that delivers unprecedented immersive gaming experiences with convincing audio and visual effects. Figure 1.1 contrasts British Legends [1], a well-known MUD that was created in the 1970s with World of Warcraft (WoW) [7], a representative commercial MMOG that was launched in 2001.

With widespread and growing use of the Internet, MMOGs have become increasingly popular since the mid 1990s. In the past few years, the total number of MMOG subscribers have been increasing on a steady exponential curve, as displayed by Figure 1.2, and a successful commercial MMOG like WoW can have more than 10 million paying subscribers [162]. This realisation has attracted both business and academic attention.

From a commercial point of view, the MMOG industry is currently one of the more successful and promising online businesses, having demonstrated prodigious profit po-

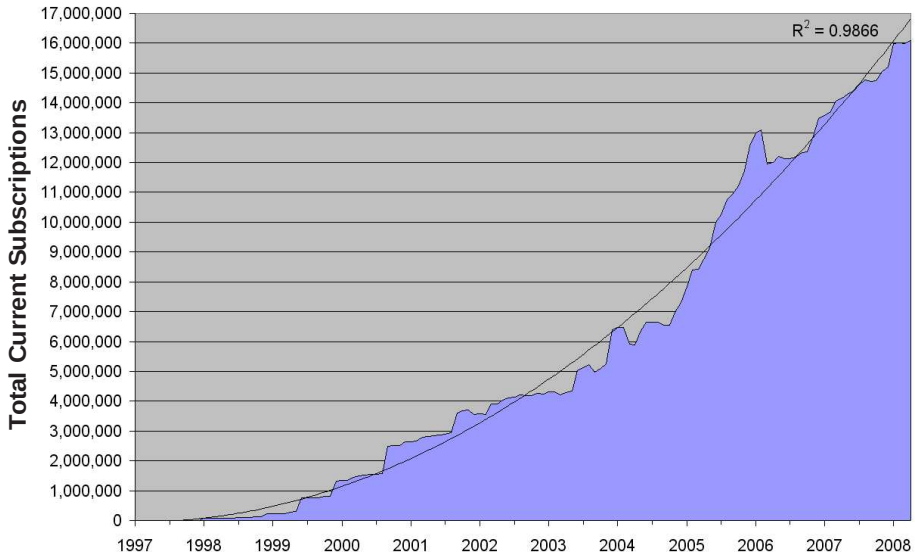

Figure 1.2: Total MMOG active subscriptions [162]

tential. At present, there are about eight million MMOG subscribers in the United States and Europe, and roughly the same number of players in Asia [162]. As shown in Figure 1.3, DFC Intelligence, a well-known video game and entertainment industry research institution, reports the growth in online game revenue which is estimated to exceed four billion dollars annually in 2009. As a result, considerable investment is being made in the design and development of new-style MMOG architectures that will mitigate the drawbacks of conventional architectures, afford higher flexibility and lower business risk for the online game industry, and thus further boost the revenue of MMOG service providers.

From an academic perspective, a MMOG is a nontrivial distributed network application type. Firstly, it features a variety of inherent complexities that deserve in-depth study. For example, a MMOG requires near real-time interactions in a highly responsive game world despite network delay; reliable, scalable and self-adapting distribution architectures; consistent, secure and cheat-proof communication protocols; robust, efficient and persistent data storage; authentication, accounting and digital rights management; artificial intelligence (AI) for virtual actors; authoring, sharing and streaming

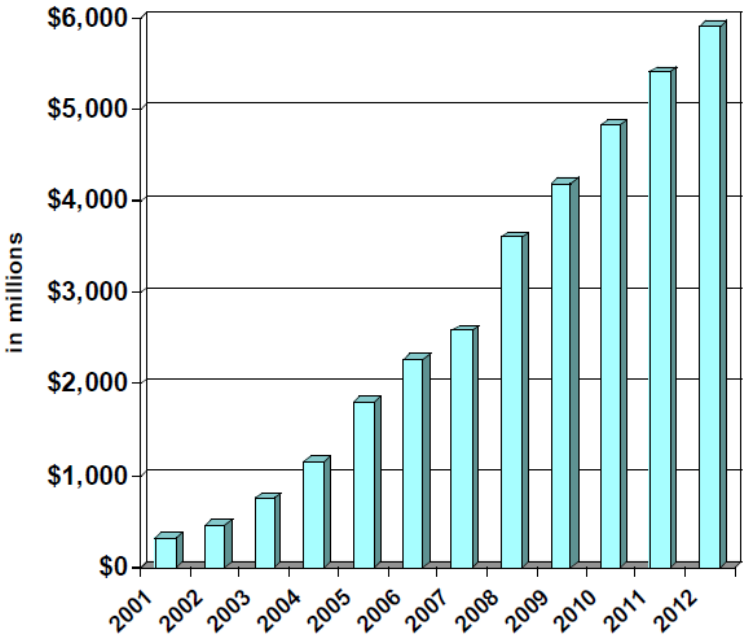

Figure 1.3: DFC Intelligence MMOG market forecast '08

of user-generated contents; mobile and ubiquitous networking; accessibility, usability and other challenging technical requirements. Secondly, significant numbers of non-game applications are also benefiting from technologies that are being developed for MMOGs. For example, military, flight, vehicle and medical surgery simulations can be carried out relatively cheaply and safely in virtual environments, and are being widely used for training and demonstration purposes. Doing such training in real life situations would be much more expensive and hazardous. Furthermore, interactive collaborations, storytelling, distance learning, e-commerce, manufacturing systems and many other applications are also taking considerable advantage of technical solutions developed by MMOG implementers. Last but not least, the open source community is keenly interested in the prospect of public MMOG infrastructures, which would make it possible for MMOG fans to design and implement their own MMOGs and run them for free.

In summary, massively multiplayer online games are evolving rapidly, from a game genre that was unknown to the public, to a prosperous industry and a significant research field at present. However, as MMOGs scale up, conventional Client/Server architec-

tures are facing various drawbacks in their reliability, redundancy, and costs of achieving scalability. The following section briefly outline the reasons behind these issues, and explains how they could be addressed by Peer-to-Peer architectures.

1.1.2 Challenges for Client/Server MMOG Architectures

To date the dominant architecture for implementing a MMOS is Client/Server (C/S) as it is relatively easy to implement and secure. In this classical architecture a centralised game server is required to host a persistent virtual world. Players, who want to join the game, must connect to this game server and constantly upload game events, such as their movements and actions. The server updates the state of the game world accordingly, and reflects the results back to every client.

Take a typical scenario in a MMOG as an example. Suppose that a group of player characters is approaching a dragon monster. Each individual player has to inform the server of its moving course and speed, and the server works out their positions and tells each of them periodically which other players are nearby. This makes a player aware of the existence of other players around itself. Next, when the players are sufficiently close to the monster, the server notifies them what kind of monster they are about to see, so that the players can render the appearance of the monster from their local media. When the players start to attack the monster, they inform the server of what actions they take, and the server executes an AI program that determines how the monster reacts during the fight. In the meantime, the server also needs to order and process the game events, to calculate how much damage the monster and players have received, and most importantly, to send the results back to every client in due course. At the end of the battle, the server updates players' accounts with additional experience, skills, loot and so on.

Obviously, in the previous example the game server plays a crucial role and has to carry out substantial computation and communication work. The situation was not too bad in

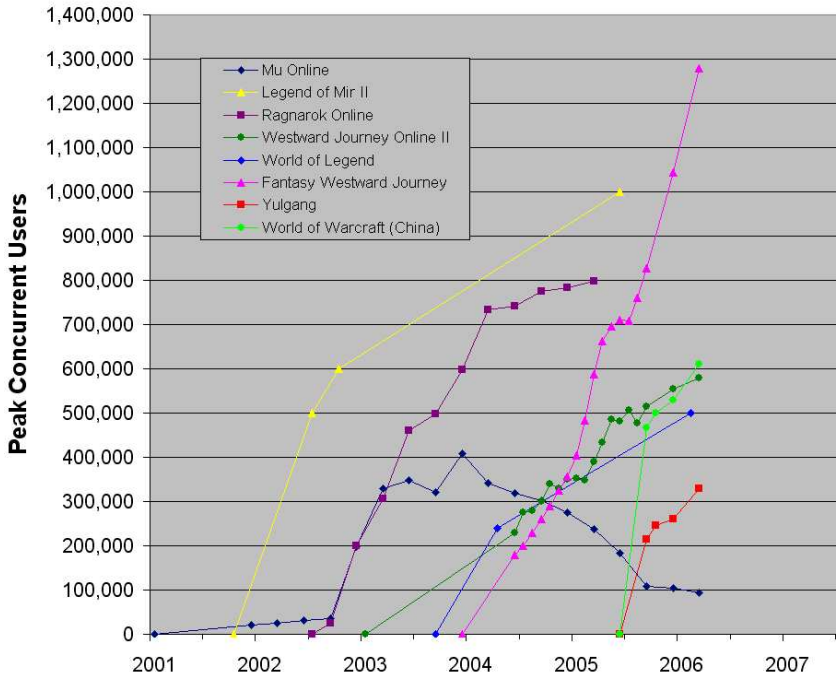

Figure 1.4: Asian market peak concurrent users [162]

the early 1990s, when a MMOG only had around 100 simultaneous players. However, today game server's scalability has become a crucial challenge because the number of players has exploded to hundreds of thousands. Figure 1.4 shows the peak concurrent users of several asian MMOGs from 2001 to 2006. Except for MU online, all the other MMOGs were scaling up rapidly, especially Fantasy Westward Journey, which achieved a six-fold increase of concurrent users within two years.

Cluster technologies [24, 25] are being used to achieve server-side scalability. For example, Google in 2003 maintained a cluster of more than 15,000 servers in order to provide their service, and in 2008 increased this to over 45,000 servers, located in a number of major cities all over the world [25]. Such facts suggest that with appropriate distributed computing and load-balancing mechanisms, C/S architectures can be made to scale to meet the requirements of massive numbers of clients. Significant research efforts have already shown how to support a MMOG with multiple game servers (Section 2.3.1), e.g. [62, 51, 47, 68]. However, the costs of achieving scalability in this way have been very considerable.

In 2004, Google spent $250M on hardware [92]. This is unlikely to be an affordable budget for most MMOG service providers. It is estimated that the cost of delivering a medium-sized MMOG for 30,000 simultaneous players using a C/S architecture usually exceeds $10M, and ongoing support costs will consume up to 80% of its revenues due to the rent of network bandwidth, dedicated machine rooms, UPS systems, cooling systems and support staff [36, 101]. In summary, while C/S architectures can be made scalable, doing so is expensive and makes supporting a large scale MMOG a demanding business project.

In addition to the costs of achieving scalability, C/S architectures also have other inherent drawbacks, including reliability and redundancy. C/S architectures are not very fault-tolerant. If a server stops functioning for software or hardware reasons, all the players supported by the server are stuck and unable to play until the server or a backup server comes online again. For example, after Blizzard's WoW game server crashed on 21 Jan 2006, thousands of players complained, which finally led to the company having to compensate those players [112]. Even worse, such accidents have happened several times during the past few years, e.g. a recent severe accident occurred on 18 May 2009, when "a sudden wave of crashes hits the north American WoW servers" [161]. An effective way of reinforcing a game server's reliability is to introduce mirrored backup servers with special synchronisation and game state replication mechanisms [51, 112, 126]. However, this further increases the costs of the game server infrastructure.

Redundancy is caused by underexploitation of hardware resources. Many well-known commercial MMOGs meet scalability requirements using a divide-and-conquer strategy that assigns game zones to different physical servers to process [7, 5, 10]. Some of the zones may be more interesting or profitable for players in terms of treasures, and thus they attract many more players than other zones which remain sparsely occupied. Such uneven distribution of game objects is referred to as a "flocking" issue [47]. One of the consequences of flocking is redundancy, as hardware resources on some game servers are not fully utilised. Furthermore, a MMOG service provider has to guarantee that the hardware configuration and network bandwidth of its game server infrastructure are ad-

equate to support a maximum number of simultaneous players at peak time. However, the average online population could be much smaller, and as a result lots of computing resources become redundant and are sitting idle most of the time. In this case, a few MMOG middleware and service platforms have been proposed to share the computing power of a single game server infrastructure among multiple game instances (Section 2.3.2), such as IBM's butterfly.net [8] and Sun's Game Server technology [54]. Unfortunately, these technologies are not able to solve the intrinsic problems of C/S architectures from the root, and they have not been widely adopted by the game industry.

1.1.3 Potential of P2P MMOG Architectures

Peer-to-Peer (P2P) MMOG architectures (Section 2.4.1) offer themselves as interesting alternatives to C/S MMOG architectures, because they exploit application participants' computing resources, and can be effectively organised to regulate the workload imposed on game servers. Recently, the computing power available on personal machines has grown dramatically along with cheap network bandwidth. This has created the opportunity to migrate major parts of a game server's functionalities to resource-rich client machines to support a MMOG at a low cost. In the previous example about players' engaging in combat with a dragon, one of the players could host the AI program of the dragon on behalf of the server. The players could also exchange game events and simulate corresponding results among themselves instead of relying on a server.

P2P architectures not only have the potential to reduce the cost of a MMOG, but can also avoid the scalability, reliability and redundancy problems of C/S architectures. A P2P MMOG is inherently scalable as the more concurrent users it has, the more computing resources it has available to support system services. Considering that most application participants' computers are more capable than just being able to run the game software, a P2P MMOG could be self-sufficient as long as a sufficient proportion of game participants are willing to contribute their spare resources to run extra tasks for the rest of the players. It would also be possible for players to dedicate their computers to a

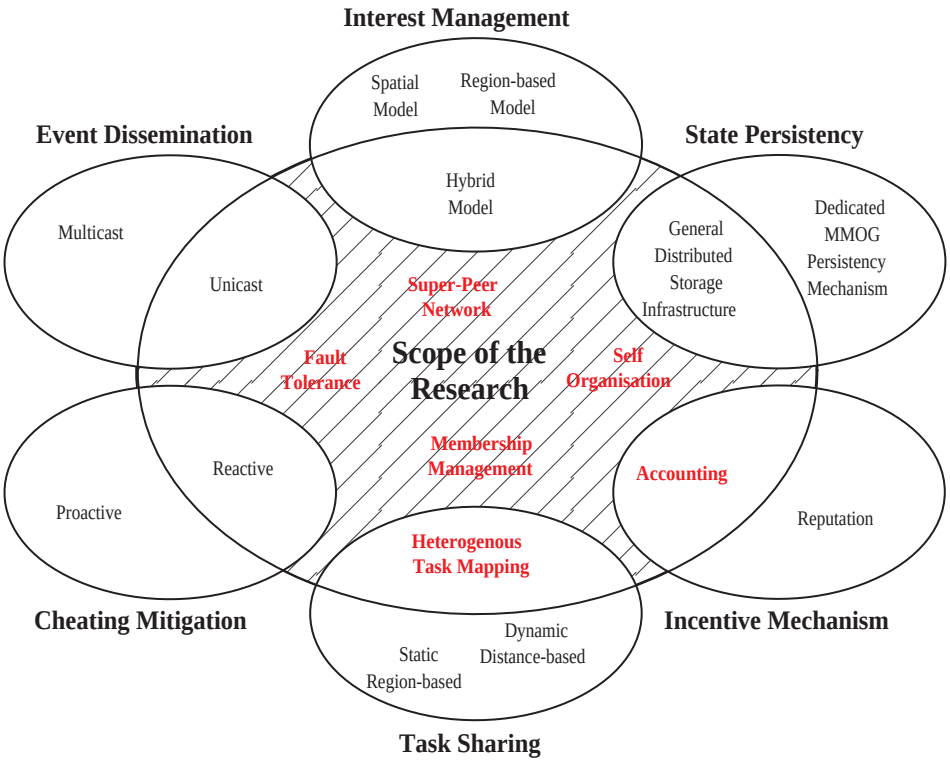

Figure 1.5: Scope of the research

P2P MMOG, even when they were not playing the game by themselves, to earn playing credits or for altruistic motives.

At a macro-level, P2P architectures are more fault-tolerant than C/S architectures. A game participant's computer in a P2P MMOG is mainly responsible for its own playing of the game, plus the hosting of a limited number of public tasks. Hence the failure of the participant's computer will only affect a few other players in a short period of time, rather than preventing normal play by a large cohort of other players in the game. When additional resources are available in a P2P MMOG, they can be employed as backups for current resource providers, which assists recovery from exceptional circumstances. Such "redundancy" has a positive effect, and it is different from idle resources that may be wasted on expensive commercial servers in C/S architectures.

In summary, P2P architectures offer many potential benefits compared with conventional C/S architectures. However, they also give rise to a series of technical challenges,

such as interest management [163, 168, 88, 140], event dissemination [120, 34, 88, 86], task sharing [167, 29, 86, 34], state persistency [144, 173, 91, 23], cheating mitigation [73, 107, 97, 115], and incentive mechanism [77, 89, 152, 116]. To adapt MMOGs completely to P2P architectures is a challenging research topic, and significant research efforts have been made in the literature, e.g. [142, 108, 87, 59, 139, 57, 79, 118, 64]. The next section outlines the scope and objectives of the research in this thesis.

1.2 Research Scope and Objectives

A major theme of this research is to present the design of a new framework for P2P MMOGs, called ***Mediator***. It addresses six key design aspects, namely interest management, event dissemination, task sharing, state persistency, incentive mechanism and cheating mitigation in an integrated system. For each of the issues, many different approaches have been proposed in the literature, as depicted by Figure 1.5. Some of them can be employed by the Mediator framework directly. For example, in the aspect of interest management an existing hybrid model [168] is employed. In Figure 1.5, such related work is labeled in black, and in contrast topics labeled in red are addressed by this research, including self-organisation, super-peer selection and fault-tolerance mechanisms (Chapter 4); an efficient membership management mechanism (Chapter 5); and a novel heterogeneous task mapping infrastructure that comes with a built-in accounting mechanism (Chapter 6).

The objectives that guide this research are to:

- Identify the problems faced by conventional MMOG architectures.
- Identify key design issues in adapting MMOGs to P2P architectures.
- Study, analyse and classify existing approaches and mechanisms that have been proposed to address each individual key issue.

- Present a general P2P MMOG design framework that addresses all the key issues in an integrated system, and relate it to existing frameworks.
- Enhance the framework with suitable self-organisation, fault-tolerance and efficient membership-management capabilities.
- Design, implement and evaluate a novel heterogeneous task mapping infrastructure that supports incentive mechanisms, in comparison with existing region and virtual distance based NPC host allocation systems.

1.3 Research Methodology

This research has been conducted using the following methodologies:

1. Survey & Critique At an early stage of the PhD work, a survey was carried out by the author to develop an in-depth understanding of the background and problems in this research field. In this process, a number of commercial and technical drawbacks of conventional MMOG architectures were identified, and previous research efforts that have been made to adapt MMOGs to P2P architectures were systematically and critically analysed. An outcome of this phase is the characterisation of six key design issues for P2P MMOGs that are depicted in Figure 1.5 and explored in Chapter 3.

2. Framework Formulation An important milestone of this research has been the formulation of a general and flexible framework that addresses the six design issues using a self-organising super-peer network built on a P2P overlay infrastructure. Considering that a super-peer's reliability and availability are not comparable with a dedicated game server, the design of the framework provides a range of fault-tolerance mechanisms (Chapter 4); an efficient membership management technology (Chapter 5); and a novel task sharing mechanism (Chapter 6).

3. System Prototyping To validate the correctness of the theoretical designs, proof-of-concept prototypes of key system components have been implemented, together with corresponding demonstration and test bed applications (Chapter 5 & 6). This practical work enables quantitative measurement and evaluation of key P2P MMOG components, and provides crucial evidence for its strengths and weaknesses.

4. Modeling & Simulation It is infeasible to carry out experiments with a realistic MMOG comprising thousands of computers and people on a real network. Hence, a simple mathematical model of the NPC host allocation process has been built and analysed using parameter values typical of real MMOGs (Chapter 6). Also, simulation plays an important role throughout this research. All the experimental results presented in Chapter 5 and 6 have been obtained with a discrete event simulator that is provided by FreePastry software package, an open source implementation of Pastry [143]. The simulator is able to simulate the routing and forwarding of messages in a large scale P2P overlay network with Internet-like communication delays among its participants.

1.4 Contributions to Knowledge

The thesis makes five research contributions:

A critical analysis of the characteristics of Virtual Environments (VEs) and Massively Multiplayer Online Games (MMOGs), their design and implementation. The review includes a critical appraisal of the Client/Server (C/S) architecture, service platforms, middleware solutions, and Peer-to-Peer (P2P) architectures. By analysing and comparing the advantages and drawbacks of these approaches, the review motivates the exploration of P2P architectures with a view to building more scalable, reliable and lower cost VE and MMOG applications. **(Chapter 2)**

An identification of key design issues for P2P MMOGs [67]. An analysis identifies six important design issues to be addressed by P2P MMOGs, namely interest management, event dissemination, task sharing, state persistency, cheating mitigation, and incentive mechanisms. Design alternatives for each issue are systematically compared, and their interrelationships discussed. Furthermore, several representative P2P MMOG architectures are classified and evaluated. **(Chapter 3)**

The design of a novel P2P MMOG framework that addresses all the key issues identified by the previous analysis within an integrated system [64]. The key idea of the Mediator framework is to distribute the functionalities of game servers to game participant machines using a self-organising super-peer network. The design of the Mediator framework is flexible and extensible. For example, while the framework provides approaches that directly account for many key issues, it can adopt alternative mechanisms or policy plug-ins to solve those issues in various ways. Furthermore, new super-peer roles can also be introduced into the framework according to identified requirements. Key components of the framework have been implemented and evaluated in Chapter 5 and 6. **(Chapter 4)**

The design and implementation of Membership-Aware Multicast with Bushiness Optimisation (MAMBO), and an evaluation of it in comparison with a conventional supervision architecture [65]. MAMBO reuses the communication structure of a tree-based application-level multicast system to track group membership efficiently, i.e. to record when peers join or leave a group. MAMBO has been implemented as policy plug-ins for Scribe [38], which entails only a few changes to the underlying technology. A demonstration application Peer-to-Peer Online Market Place (POMP) has been implemented and evaluated to show that MAMBO is more scalable than a conventional C/S architecture, and incurs less communication overhead. The Mediator framework is designed to use MAMBO to establish a supervision infrastructure that enhances the dependability of its super-peers. **(Chapter 5)**

The design, implementation and evaluation of Deadline-Driven Auctions (DDA), a novel task mapping infrastructure for NPC host allocation in P2P MMOGs [66]. The novelty of DDA lies in its functioning as a dynamic task mapping mechanism for heterogeneous P2P environments, which is different from existing region based and virtual distance based NPC host allocation approaches. A test bed application has been implemented. Both experimental and analytical results demonstrate that DDA provides four significant advantages. Firstly, it is self-organising as its infrastructure can be automatically assembled and managed. Secondly, it efficiently allocates computing resources for large numbers of real-time NPC tasks in a simulated P2P MMOG with the better part of 1000 players. Thirdly, it supports gaming interactivity by keeping the communication latency among NPC hosts and ordinary players low. Finally, it supports flexible matchmaking policies, and with a friendly incentive policy, can establish a cooperative economic model that helps motivate application participants to contribute their resources to the system. Though DDA is primarily designed to support the distributed hosting of NPC objects in P2P MMOGs, it could also support general P2P applications that feature computational and interactive tasks with various deadlines. **(Chapter 6)**

— If I have seen further, it is by standing on the shoulders of giants.

Isaac Newton

Chapter 2

Background

2.1 Introduction

This chapter presents a critical analysis of the characteristics, design and implementation of virtual environments and massively multiplayer online games. In order to set a scene for the discussion, the definitions for the target applications are clarified (Section 2.2.1), their typical elements (Section 2.2.2), themes and goals (Section 2.2.3) are explained, and their general requirements are outlined (Section 2.2.4).

Client/Server architectures have been predominantly employed for conventional MMOGs, because they are relatively easy to implement and secure. Though these classical architectures convey some benefits, they suffer from significant technical and commercial drawbacks as they scale up (Section 2.3.1). On the other hand, various forms of MMOG middleware and service platforms also have been proposed in the literature (Section 2.3.2). However, such alternatives are by nature shared game server clusters, hence they are not able to overcome inherent disadvantages of C/S architectures.

In contrast, Peer-to-Peer architectures seem to be more attractive by providing a scalable and low cost way of building MMOGs (Section 2.4.1). Currently, many well-known

application types have been successfully deployed in the P2P fashion (Section 2.4.2), especially with the support of mature P2P overlay infrastructures (Section 2.4.3). The last part of this chapter summarises the advantages provided by P2P MMOGs as the motivation for this PhD research, as well as identifying a number of challenges that must be overcome (Section 2.5). Later in this thesis, Chapter 3 systematically interprets these challenges into six key design issues for P2P MMOGs, Chapter 4 presents a framework that addresses the issues within an integrated system, and Chapters 5 and 6 further elaborate on significant components of this framework.

2.2 Virtual Environments and MMOGs

2.2.1 Terms and Definitions

A Virtual Environment (VE) is a computer simulated environment for its users to inhabit and interact via avatars [12]. This inhabitation is usually achieved with two or three-dimensional graphical representations of humanoids, or other graphical or text-based avatars. A VE is also referred to as a CVE (Collaborative VE), DVE (Distributed VE), IVE (Immersive VE), MUVE (Multi-User VE), or NVE (Networked VE). Recently, VEs are increasingly being used in a variety of contexts, e.g. military simulation and training [18, 96, 154], educational and research platforms [30, 85, 169], e-commerce [9], leisure [13] and for other social purposes [114, 103, 80]. Figure 2.1 shows a virtual shop and business conference in SecondLife.

VEs' potential uses in computer game applications have been exploited since the early 1990s, which have resulted in a plethora of MMOGs nowadays. As a complex application genre, many different interpretations have been provided in the literature to define a MMOG. Central among them is the idea that a MMOG is ***a type of computer game that enables thousands of players to interact simultaneously in a persistent game world when they are connected via a network such as the Internet***. MMOGs

Figure 2.1: E-commerce and business conference in VE SecondLife

can be further categorized into several sub-types, such as MMO Role-Playing Games (MMORPG), MMO First-Person Shooter games (MMOFPS), MMO Real-Time Strategy games (MMORTS) and so on. Because the first and most popular type of MMOG is the MMORPG, many people equate a MMOG with a MMORPG, even though the more general concept emerged later. In this thesis, the term "MMOG" is used to refer to a MMORPG. Well-known examples include Ultima Online [14], EverQuest [5], and World of Warcraft (WoW) [7] which is depicted by Figure 2.2.

2.2.2 *Essential Elements of a MMOG*

There are several essential elements in a MMOG. The most immediately noticeable ones are a persistent game world, considerable numbers of player avatars, non-player characters, player artifacts, buildings, plants and terrain features.

Figure 2.2: Screenshots of a popular MMOG - World of Warcraft

Persistent Game World

A MMOG features a persistent virtual game world, which is a computer-based artificial world of 3D spaces that provide a setting for fantasy gaming scenarios. For this reason, a MMOG is sometimes directly called a Persistent World (PW) [95]. The persistency comes from maintaining and developing the state of the gaming world around the clock - a game world is always available to its users and world events happen continually [53]. Unlike other game genres, a MMOG's plots and events continue to develop even while some of the players are not playing their characters online.

In order to accommodate large numbers of concurrent players, a game world usually has a vast territory. This makes it necessary to divide up the whole game world into multiple sub-spaces, so they are easier to manage and maintain. The largest granularity of such sub-spaces are named "game zones", which play an important role in the underlying implementation of a MMOG. At present, many well-known commercial MMOGs

have adopted a zone-based design that assigns game zones to different physical servers to process [7, 5, 10]. In this way, when a player's avatar moves between zones, the player's connection is redirected from one game server to another. More details about conventional MMOG architectures are discussed in Section 2.3.

Player Avatars

Players in a MMOG are visually represented by avatars that are rendered by client side software. Specifically, a player may be denoted by two kinds of avatars - a pilot avatar present on the player's own machine, and a drone avatar which apes the pilot avatar on all the other machines connected to the world. In the communication layer, a pilot avatar is controlled directly by the user, whereas a drone avatar requires its state and behavior updates to be delivered over the network, which introduces a latency issue.

Communication latency is the amount of time required to transfer data from one place to another, and in MMOGs latency exhibits itself as delays when two players are interacting. A number of factors contribute to the latency. For example each cable modem involved in the communication typically adds 30-40 ms to the transfer time, and Internet routers can add hundreds of milliseconds due to their caching of data before forwarding. The situation is even worse when overloaded routers decide to drop data, which can introduce penalties in the 400-500 ms range for protocols like TCP to detect lost data and resend it. Even in an ideal communication channel, we still have to take the ultimate limitation, the speed of light, into consideration. For example, a message sent from the East Coast to the West Coast of the United States will take about 20 ms to get there just due to that limitation[53].

MMOG designers often incorporate a latency tolerance of 250 ms into the communication model by employing various approaches for estimating information about other players in the game, e.g. a player avatar's current position can be anticipated according to its previously determined position, velocity, course and elapsed time. The most widely adopted approach is bucket synchronization with Dead Reckoning (DR) [26], which is

a part of the Distributed Interactive Simulation (DIS) [2] and High-level Architecture (HLA) [6, 105, 49] standards.

Non-Player Characters

While player characters (PC) are represented by player-controlled avatars, there are also AI-controlled non-player characters (NPC), which are important virtual actors that drive continuing storylines in a MMOG. Figure 2.3 illustrates two typical classes of NPCs. On the left are the Hippogryphs in WoW. These creatures are able to carry PCs along with them on their patrol routes, and provide a means of transport in various areas of the game world. On the right, a screenshot depicts a group of PCs fighting against an Anubisath Sentinel in WoW. Anubisath Sentinels guard the Temple of Ahn'Qiraj and usually come in packs of four. Only by defeating two packs of them will the players encounter the first boss, the Prophet Skeram.

It is necessary to distinguish the NPCs above, because they have different requirements for network communication. In order to provide a shared sense of space among players, each player must maintain a local copy of the game state. Sometimes, when one player performs an action, the game state of all other players affected by that action must be updated immediately by real-time gaming events. If the transmission of such events is seriously delayed by the network (i.e. the latency problem discussed in the previous section), it will result in inconsistent game states at different player machines. Fighting against Anubisath Sentinels is an example of real-time interactions, in which the game states of multiple players must be tightly synchronized. However, hiring a Hippogryph is an example of a latency free interaction, because it has little effect on other players' game state, and it does not much matter that other players perceive such an event at a different time. Conventionally, an interaction between two PCs is called a Player-vs.-Player (PvP) interaction, and an interaction that involves both PCs and NPCs is called a Player-vs.-Non-Player (PvN) interaction. This thesis focuses on real-time PvP and PvN interactions, because they have higher Quality of Service (QoS) requirements, and thus

Figure 2.3: Typical NPCs in a MMOG

need special technical support and investigation.

A typical MMOG must supply its game world with large numbers of NPCs as required by game scenarios. The creation of a NPC is referred to as "spawning", and respawning means the recreation of a NPC after its death, destruction or removal. Either space-based or timer-based approaches have been used to respawn NPCs in a MMOG. A space-based approach creates and scatters NPCs all over the game world according to a predefined density, i.e. number of NPCs per space unit. In contrast, a timer-based approach respawns NPCs periodically, and attempts to keep the number of NPCs and PCs to a stable ratio. Moreover, a dynamic timer can be applied to improve further on the performance of a respawning algorithm. For example, when a big crowd of players are all killing dragon NPCs together, dragons may die out after one minute on average. If a static timer of ten minutes was adopted, the players would have to wait for nine minutes before more dragons are respawned. In this case, a dynamic timer is helpful to adapt the respawning interval appropriately. In practice, dynamic timer-based respawning mechanisms offer better gaming experiences, so they have been widely used in popular commercial MMOGs [7, 10].

Last but not least, with regard to the creation of NPCs, the term "spawn point" should be explained. A spawn point in a MMOG is not a single point, but an area of the virtual world, in which a random point is selected as the initial position of a NPC. Usually,

spawn points are arranged according to in-game logics, e.g. Orcs are only respawned in a wood, whereas Mermaids are only respawned in the sea.

Traditionally, NPCs are hosted by a central game server, consuming significant processing power and network bandwidth. The understanding of PCs, NPCs, and their relationships and behaviours is crucial for proposing novel MMOG architectures, as well as for the modeling and simulation of MMOG game sessions.

Other Objects, Entities and Items

Besides the PCs and NPCs, there are also considerable numbers of other objects in the game world. Some objects are immutable, such as buildings and street fittings, but many objects are mutable entities and items, such as weapons, equipments, magic potions and gemstones. Many mutable objects are dropped by defeated monster NPCs, as the loot for the victors. A player is able to pick such items up from the ground, put them into their own inventory, consume them, or trade them for virtual currency.

Uneven distribution of game objects often leads to a flocking issue in a MMOG, which means the movement of many players to one area (i.e. hotspot) of the game world [47]. Flocking occurs because it is inevitable for a game world to have some regions that are more interesting, or simply more profitable for players in terms of treasures. Experienced players tend to "camp" at certain places (referred to as a placeholder), waiting for rare and valuable items to be dropped by a NPC. One of the consequences of flocking is a serious load-balance problem, since such regions attract many more people than other regions which remain sparsely occupied. More details about load-balance for game servers will be discussed in Section 2.3.

2.2.3 Themes and Goals in a MMOG

The majority of popular MMOGs are based on traditional fantasy themes involving quests, monsters and loot. However, unlike in other game genres where a player is

required to complete tasks in a sequence, usually there is no set goal in a MMOG. In other words, players are set down in a game world free to do what they will. There is no overarching goal to "beat the game" in the end. Most players settle into a pattern of developing their characters, and nearly all MMOGs endeavor at providing distinctive character progression systems. Traditionally, players earn experience points to make their characters more powerful by completing quests at the behest of NPCs. These often involve combat with monsters, and offer opportunities to acquire wealth in terms of virtual currency, rare weapons and other valuable items.

Recently, there has been a trend among MMOGs to support in-game guilds or clans, which in turn enable various forms of collaboration and teamwork that require the synergy of tens of players. So, it becomes increasingly difficult for an individual player to survive and develop itself in the game world without assistance from other players. Considerable research effort has been put into study of the social side of a MMOG [61, 30, 128, 147], and more details about the impact of players' social behaviours on the implementation of a MMOG will be discussed in Section 2.3.1.

2.2.4 General Considerations in a MMOG

As discussed above, a MMOG is a mixture of various game genres, and it features significant levels of interactivity and complexity. In general, the following issues must be considered when designing and implementing a MMOG:

- Consistency - In a MMOG, all users should see the same sequence of events in the same order, regardless of the underlying architecture or implementation. For instance, player A may shoot player B when B is shooting player C. If A's shooting action occurred prior to B's, then B should be dead first and thus C is not hurt. However, the problem is that the events may have occurred on geographically separated machines and event details must be sent between the player's machines by message passing. This implies that a mechanism like time-stamping of events

must exist on different machines with clocks that are globally synchronized. A MMOG must provide consistent event delivery functionality in order to insure that A, B, C and all the other users in the vicinity experience and see the same event sequence.

- Real-time - In a MMOG, when an event occurs at time t for one user, other users should see that event at time t as well. Real-time interactions are especially vulnerable to latency impacts. Therefore, the QoS of network performance is important to a player's gaming experience, which in turn determines the business success of a MMOG project. Support for adequate QoS puts a heavier burden on a MMOG's bandwidth requirement, e.g. several gigabytes per hour, so a MMOG is widely accepted as a bandwidth intensive application.

- Scalability - As indicated by its name, a MMOG enables massive numbers of players to interact with each other simultaneously, so a valid MMOG architecture should be readily scalable. In addition, the actual online population may vary significantly over time. Though the architecture is able to support the maximum number of players during peak time, it may waste considerable amount of computing power during off-peak time. To support a MMOG with as few resources as possible, and to make the application highly scalable is a major challenge.

- Security - Security issues present themselves at all stages of the design and implementation of a MMOG. Attention must be paid to the security issue, because outside the game world any user is a potential cheater or cracker aiming to take down a game application or generally ruin the economics of a game society, and inside the game world we can not assume any norm will govern a player's behaviour.

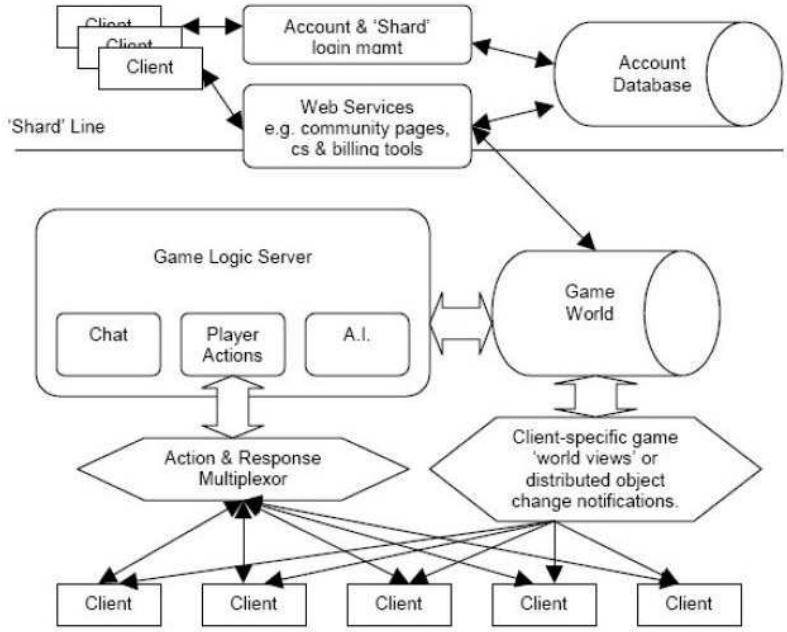

Figure 2.4: A highly simplified view of the C/S architecture for a MMOG [95]

2.3 Conventional MMOG Architectures

2.3.1 Client/Server Architecture

A single game server

The C/S architecture has been the predominant paradigm for implementing traditional MMOGs. As shown in Figure 2.4, in a C/S architecture all players send their updates to a server, and then the server updates the state of the game world accordingly and reflects the result back to every client. In other words, in an N-player game, a maximum of 2N messages are exchanged between the server and the players in each round. If any one player suffers from a poor connection, that will only impact on the player's own game, and to a limited extent on the games of players that attempt to interact with them.

A game server's functionalities include (but are not limited to):

- To perform administrative and control tasks, such as accounting for time spent in the game on an individual basis.
- To receive input commands from each client, turning key presses into actions like jumps and weapon fires, and to send updates to concerned clients.
- To keep track of entities such as avatars, monsters, bullets, etc., and to calculate the position of moving objects using physics models.
- To maintain the game world's parameters, such as gravity, lighting, and map information.
- To communicate with client side programs reliably.
- To reduce message traffic to individual users by not sending packets to those users if the packet in question is out of the area of interest of the potential packet recipient.
- To compress multiple packets into a single message to eliminate redundant message flow and reduce communication overhead.
- To convert packet bursts into smoother packet rates, thus delivering packets at a slower rate than they are generated by individual users.

In this architecture, both computational and networking workloads are offloaded onto the game servers. The situation was not too bad in the 1990s, when the word "massively" only implied around 100 simultaneous players. Currently it has deteriorated to become a real crisis because the number of players has exploded to hundreds of thousands [162]. Though game developers keep trying to optimize their design and to minimize the message traffic, a single game server is still too underpowered to support a medium-sized MMOG by itself. A popular recourse to address this problem has been to disperse the server load onto a group of game servers.

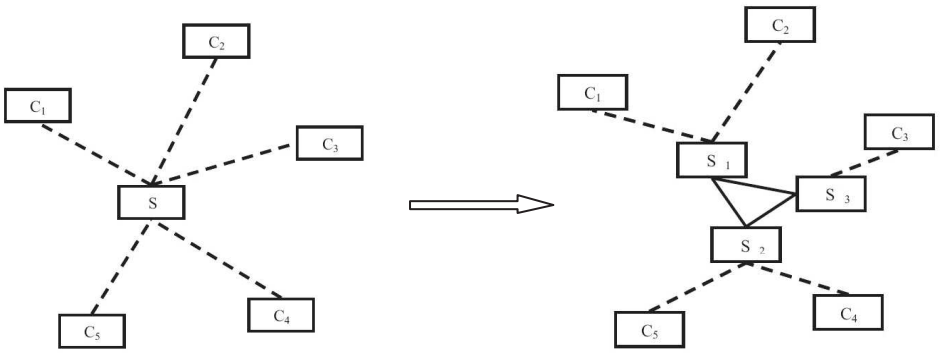

Figure 2.5: From a single server to distributed server clusters

Server clusters

A game zone is the largest granularity of sub-space in a virtual game world, as discussed in Section 2.2.2. It provides an explicit way for a MMOG to partition a huge game world into smaller pieces that can be conveniently mapped to separate servers. Figure 2.5 depicts this approach, in which multiple game servers are introduced, sharing the total workload. However, special synchronization mechanisms must be applied to connect these servers seamlessly [62, 51, 47, 68], so as to maintain the consistency of the game state. A game server cluster is also referred to as a game server farm, and Figure 2.6 shows Sony's game server farm that supports its famous MMOG, EverQuest2 [5].

Pros and cons of the C/S architecture

It has been argued that only a C/S architecture is appropriate for MMOGs and that all MMOGs should be constructed using it [69, 19], as it provides the following significant advantages:

- Authority - In the C/S architecture, the global game state is managed and stored in a centralized way, which confers the advantage of giving game providers full control over their games [121].

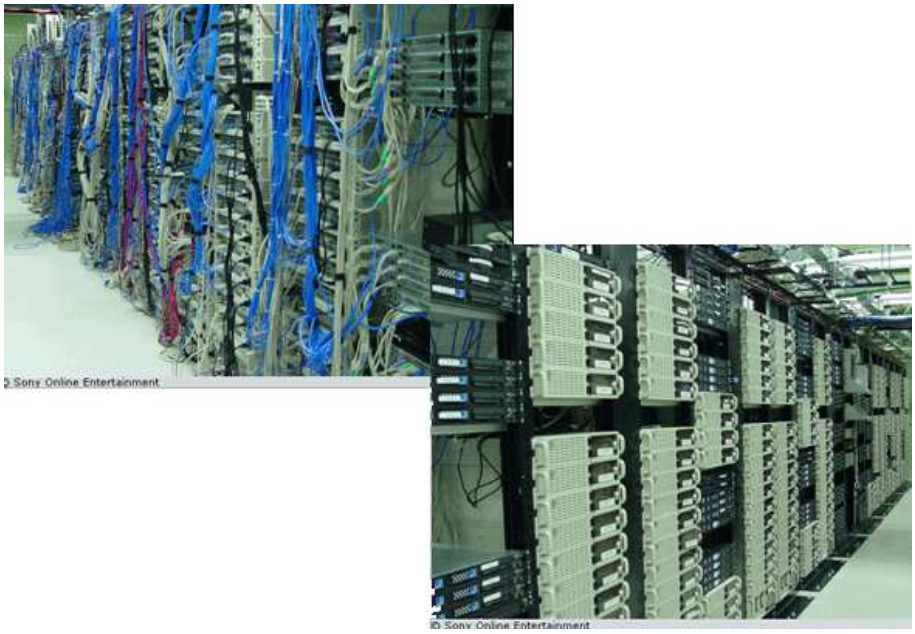

Figure 2.6: Sony's EverQuest2 game server farm in United States

- Security - Since the game server has sole authority to validate every action request sent by clients before carrying them out, it has good prospects to prevent cheating [97].
- Simplicity - The C/S architecture is, at least compared to P2P, straightforward to implement [127]. It is widely accepted that a MMOG is hard to design, build, test, and support. Hence few implementers may want to take the risk to "make trouble out of nothing".

However, just as a coin has two sides, the C/S architecture also exhibits various drawbacks, which undermine its ability to support large scale, sophisticated, interactive multiplayer online games [149, 87, 136].

Reliability The C/S architecture is not inherently fault-tolerant. Zone-based MMOGs can experience what is known as a "zone-crash", where the server supporting that region of the game world stops functioning. When this happens, all players in that region are stuck and can not play until the server comes back up.

Traditionally, game servers have only supported limited persistence, which means that only a player's state is stored persistently to discs from the memory image, whereas a region's inventory (the objects lying on the ground) and dynamically spawned monsters will be wiped out through a restart. The granularity for disaster recovery of a player's state may vary from a couple of minutes to several hours.

Scalability Scalability is a multi-faceted issue with accurate estimation of hardware performance as one problem, and accurate estimation of the online population as another. Commonly, it is hard to predict how many players a hardware platform can support, and how much network bandwidth is enough to ensure a comfortable gaming experience. The only assured way to make such a decision is to carry out real experiments. So, before any commercial MMOG project is officially launched, a series of public tests need to be performed, which may take several months.

Furthermore, because on the one hand most MMOG projects employ the pay-for-play business model based upon a monthly fee, and on the other hand most players also pay for their Internet connections monthly, it becomes flexible for players to choose their favorite time to play online. In this case, although MMOG service providers can calculate how many players have subscribed to a MMOG in total, it is impossible to forecast accurately the number of simultaneous players at any one time. Underestimating numbers may result in major customer dissatisfaction due to poor availability and performance of the game, while overestimating means wasted money due to underexploited idle resources.

Redundancy C/S architectures struggle to deal with flocking which is discussed in Section 2.2.2, because it is hard to achieve dynamic load-balancing among all game servers. As a result, in zone-based MMOGs, all region servers are designed to be equally capable to host a maximum number of players at peak time, though in most cases these computing resources are redundant and sitting idle.

In addition, even for zone-based MMOGs, each game zone still has upper limits on how many players it can support, which is referred to as the "fire marshal problem" [101]. Existing practice to solve this problem is simply to clone the entire game and launch several duplicates at the same time to accommodate large numbers of players all over the world. Sony was the first to adopt this strategy and named each duplicate a "Shard" of a MMOG (the term appears in Figure 2.4). As a business solution, shards offer a few non-technical advantages. For game developers, once a MMOG is launched, there will be little opportunity to suspend the persistent world and make changes to it. However, bugs that have been identified in an old shard can be fixed in newly launched shards. Also new scenarios, zones, character classes, equipments, and other patches and amendments can be added to a new shard. In this case, each shard is not just a clone of the original game, but is an upgraded implementation and compatible extension of the original game. Game players, before they start playing, can choose to establish their characters in the nearest shard to minimize the network latency they will experience.

However, all the advantages mentioned above cannot counteract the most serious disadvantage - shards worsen the redundancy problem, because while duplicating the game, the wastage of hardware resources is also duplicated.

Cost It is widely accepted that MMOGs are expensive - preparing to launch a MMOG nowadays takes 2 to 3 years and about 10 million dollars, and ongoing support costs will consume up to 80% of its revenues [36, 101]. MMOGs are both processing power intensive and networking bandwidth intensive in nature. It is estimated that the cost of acquiring servers for 30,000 simultaneous players (a medium-sized MMOG) is about 800,000 dollars, excluding the rent of dedicated machine rooms, UPS systems, cooling systems and dedicated support staff. The bandwidth cost for the same number of players is about 100,000 dollars per month [127]. Apparently, to offer a MMOG service using a C/S architecture is a demanding business project.

Sociodynamics In order to protect the economics of each newly launched shard, game service providers lock every player's account to the player's initial shard (as a service policy), which means that the player must abandon all their treasures accumulated in other shards if the player switches to a new shard. This tradeoff is so unsatisfactory that everyone is reluctant to switch to a new shard, and thus leads to a deep impact on the social side of MMOGs. It is common for people in MMOGs to form long-lasting friendships, which often continue as the players move on to new games. But players who play on a fully subscribed shard have no incentive to encourage friends to play the same game - the friend cannot join the player's shard, and the player loses too much to be willing to switch to a new shard which the friend can join.

In response to these issues, several novel architectures have been proposed as substitutions to the conventional C/S architecture, among which middleware and service platform technologies are discussed in the next section, and an emerging trend for P2P MMOGs in Section 2.4.

2.3.2 Middleware and Service Platforms

Considering the difficulties and cost of designing and implementing a MMOG, it is argued that [95]:

- Technical problems of creating a MMOG are subtle and complex, and largely lie outside the traditional area of expertise for game developers.
- For most game designs there is a large set of common problems that a single well-designed architecture might support.
- Game developers should focus upon game design challenges, while leaving the technical and business challenges to standardized MMOG middleware and infrastructure providers to aggregate customers effectively and achieve substantial savings.

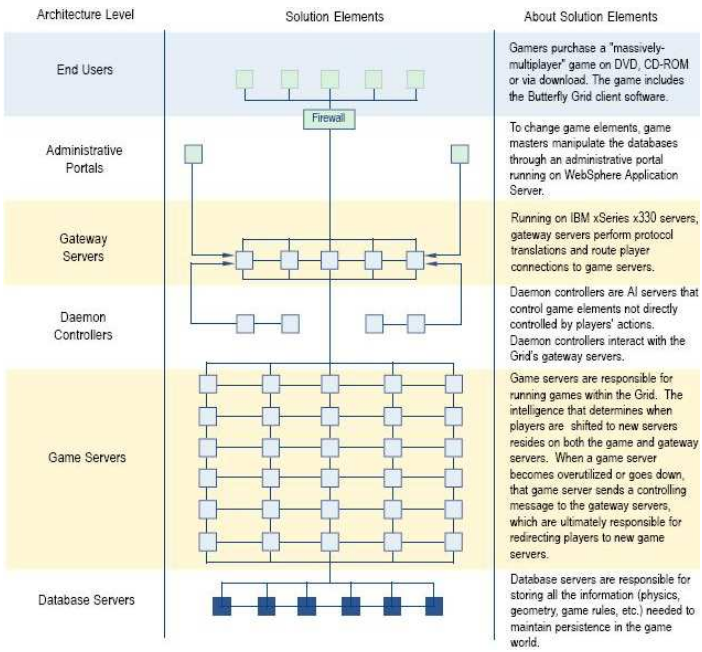

Figure 2.7: Basic architecture of the Butterfly Grid

In this case, there are an increasing number of companies offering MMOG-specific middleware solutions. These solutions can be classified into at least two categories: Grid based distributed architectures like IBM's Butterfly.net [8], and object request brokers like Sun's Game Server technology [54].

Butterfly.net

IBM's Butterfly.net project is also known as the Butterfly Grid, which is illustrated by Figure 2.7. An IBM Grid would typically be composed of two clusters of 50 IBM xSeries servers. The GRID's principal components - specialized game servers and database servers - are fully meshed over high speed fibre-optic lines, enabling the transparent routing of players to different servers in the Grid.

In the course of a game, each server will communicate (e.g. multicast) to all the other servers in the Grid in real-time. Under this peer-to-peer networking approach, players are transparently routed to the optimal server in the Grid that enables server resources

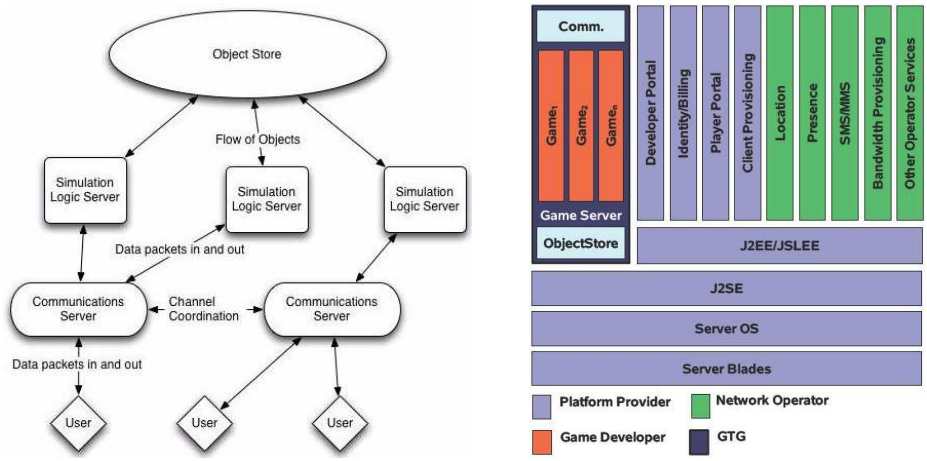

Figure 2.8: Game Server technology built upon J2EE and J2SE platforms

to be allocated to the most popular games. IBM claims that, over two years, a MMOG deployed using the traditional C/S model can generate a profit of 1.6 million dollars on subscription-based revenues of 24 million dollars. Delivering the same game over a Grid-based infrastructure like Butterfly.net would in IBM's opinion generate a profit of 12.8 million dollars, which would be an eight-fold increase in profitability [8].

Game Server Technology

Sun's Game Server technology logically is divided into 3 layers: Communications, Simulation Logic, and Object Store [54, 101], as shown on the left part of Figure 2.8.

Firstly, the Object Store layer contains the game states for all games running in the Game Server. It is an efficient (e.g. tenths of a millisecond per operation), scalable, and fault-tolerant transactional database layer that provides deadlock proof access to the simulation objects, which can either be locked (i.e. write-lock) or peeked (i.e. non-repeatable read).

Secondly, the Simulation Logic layer is responsible for executing the actual game code. Here, tasks are created based on incoming events, which in turn check objects out of the

Object Store as needed. When a task is completed, the object is updated and returned to the Object Store.

The Communications layer organizes player communication into channels of grouped communicators. It manages routing of data packets among the players and the Simulation Logic servers, and among the players themselves. It is also responsible for translation to and from other forms of networking, e.g. HTTP communications to and from cell phones.

The games industry is not a traditional market for Sun. However, the last few years have demonstrated Sun's increasing commitment to gaming. Together with Game Server technology, Sun has released abundant Java extension libraries, such as Java 3D, the Java Media Framework, Jini, JAXP and JXTA, in order to make J2SE a favorable game programming platform [54, 53, 156].

Current status of MMOG middleware

The game industry has been slow to adopt middleware solutions, for the following reasons:

- None of the middleware providers have indeed shipped MMOGs, so they are less likely to gain the confidence of a publisher or established developer.
- To develop and operate a MMOG effectively, many developers feel that they must have the in-house expertise necessary to develop one from scratch.
- A company which can afford the marketing budgets needed to create a success can also afford internal development teams instead of middleware.
- Game developers are especially prone to the "not invented here" syndrome. This seems to stem from the belief that technology is a key differentiator in the game industry.

- A company can expect to support a MMOG for many years, and therefore reliance on an external provider for fundamental technology leaves the company vulnerable on a business critical issue.
- Middleware solutions would achieve much of their significant savings of hardware resources when multiple MMOGs are hosted together, sharing the computing power of a single service platform. However, they are vulnerable to "resonance", when all the MMOGs reach their maximum number of concurrent subscribers during the same peak time.

Due to the factors listed above, the official website of Butterfly.net has been permanently closed, and there is no specific date when Sun's Game Server will be brought to the marketplace.

2.4 Rise of the Peer-to-Peer Architecture

2.4.1 Peer-to-Peer Architecture

Peer-to-peer, or P2P is not a new concept. It has existed since the Internet was taking off in the 1970s [159]. Mail servers, USENET news servers and domain name servers are widely used services that operate in a P2P fashion. Today, many factors such as the explosion of connected devices, the rapid increase of affordable bandwidth, continuous growth in computing power and storage capacities, and the proliferation of information at the edge of the network, make P2P practical for a wide number of applications.

It is useful to characterise P2P systems initially by stating what they are not. In short, P2P is not about eliminating servers. It is not a single technology, application, or business model. It should not be characterized strictly by its degree of decentralization [33]. The common characteristics of today's P2P systems include most of the following:

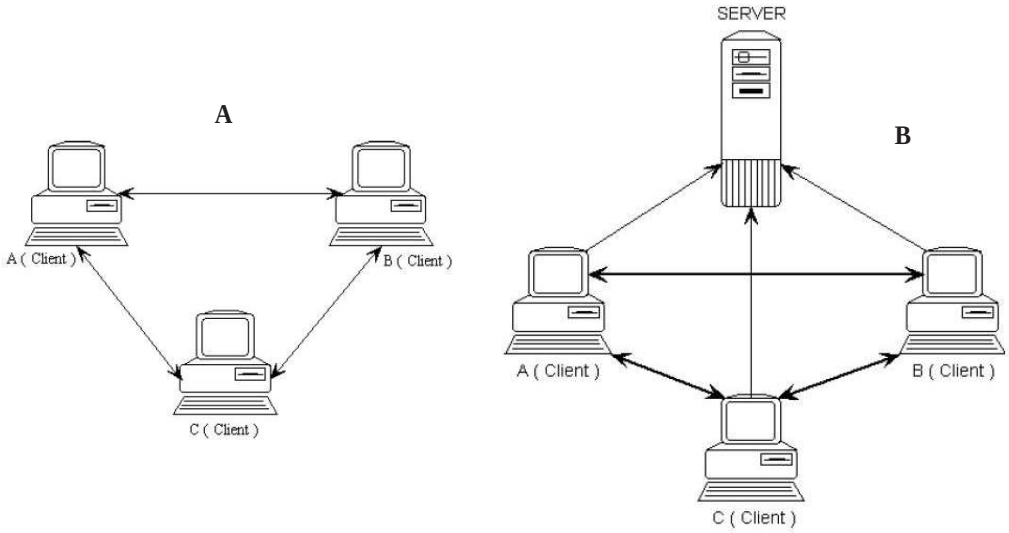

Figure 2.9: Peer-to-Peer models

- Peer nodes have awareness of other peer nodes.
- Peer nodes create a virtual network that abstracts the complexity of interconnecting peers despite firewalls, subnets, and lack of specific network services.
- Each peer node acts as both a service consumer and a service provider.
- Peer nodes form working communities of data and application that can be described as peer groups.

P2P models can be further divided into the following categories [159]:

Pure P2P As shown by Figure 2.9 part A, in the pure P2P model, the entire communication occurs among connected peers without any assistance from a server.

P2P with a Discovery Server As shown by Figure 2.9 part B, such a P2P model contains a server that is restricted to providing the names of already connected peers to the incoming peer, which notifies the server about its presence by logging in. The server only assists peers by providing a list of connected peers, but to establish connection and

communication still remains the job of the peers. Such a P2P model surpasses the pure P2P model by increasing the chances of finding a larger number of peers on the network.

P2P with Discovery, Lookup and Content Servers In this model, a server is used to provide the list of connected peers along with the resources available with each of them. Hence, this model reduces the burden on peers, as there is no longer a need to visit each peer personally for the required information. Some servers can even cache the contents that are exchanged among peers, which further improve the efficiency and availability of the whole system.

2.4.2 Peer-to-Peer Applications

P2P usage scenarios can be classified into three main categories [99] - content sharing, hardware resource sharing, and collaborative computing and communications. Typical P2P application examples may include:

Consumer file sharing Applications, such as Gnutella [100], eDonkey [155], Napster [28] and BitTorrent [94], are widely used to share media files, which in many instances violate existing copyright. Whatever one's view about current copyright laws, the power of P2P for file sharing cannot be overlooked. Many popular file sharing applications involve some degree of centralization, but vary in their implementations. At one end of the spectrum is the comparatively decentralized Gnutella, and at the other end is the original Napster that was the most centralized because it relied on a centralized server to index the content of peer nodes.

Content distribution networks Applications like OpenCola [50] and Blue Falcon Networks [15] have been developed for P2P content distribution. They are different from P2P file sharing in that the files are replicated among peers, whereas the content is distributed from a single entity, such as a newspaper, to the peer. In this case, the content

publishers no longer need high volume and high-cost web servers, because only the first copy of the content needs to be fetched by a peer, and after that peers will replicate the content among each other.

P2P network computing P2P network computing, represented by a series of well-known @HOME projects [20, 148, 35], perform computationally intensive simulations by utilizing unused domestic machines over the Internet. A significant common characteristic in such applications is to break apart infinite workloads into units that are small enough for each participant to complete in a reasonable time. Each unit then is processed concurrently with very little interaction, so such applications are also called "embarrassingly parallel computation".

Instant messengers Instant messenger applications are often designed to be P2P to embody the feature that peer nodes directly interact with each other. Popular examples of instant messengers include MSN Messenger [4], ICQ [75] and Jabber [145]. They mitigate P2P interaction by relying on a presence service from a centralised server.

Collaboration and white boarding Collaboration comprises a broad category of applications that are P2P in nature. In a work group, collaboration involves sharing of ideas and other resources. For example, team members may exchange messages or documents with each other via Email, video conferencing, and text chatting applications. During meetings, it may be helpful to have a shared virtual whiteboard to facilitate communications among distant collaborators. P2P enables all of these to happen without reliance on a central server.

2.4.3 Structured Peer-to-Peer Overlay Networks

Without centralized organization or control, it is inconvenient for a user to find a specific resource in a large scale P2P network. Traditionally, a P2P system relies on flooding

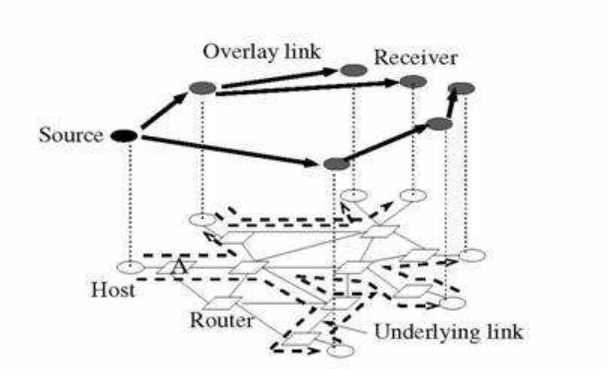

Figure 2.10: An P2P overlay network on top of a physical network

mechanisms as the only way of resource lookup, in which a peer fans out its query to all known neighbouring peers, which in turn decide whether to reply to the query, or further forward the query to their own neighbors. Unfortunately, such mechanisms do not scale well, and it is not guaranteed that a useful resource can be finally located. Two problems with flooding-based approaches have been pointed out [60]:

- Resource location and network topology are uncorrelated - available resources might not be accessible to all network nodes, and a query hit cannot be guaranteed even if the target node is connected to the network.
- The network is randomly connected - queries on an unstructured P2P network tend to have lookup complexity of the order of N hops, with N being the number of network nodes.

In order to deal with these problems, a number of structured P2P overlay infrastructures were proposed in the early 2000s, e.g. CAN [134], Pastry [143], Chord [150] and Tapestry [175]. These infrastructures organize peers into a logical topology, according to the contents, resources, or services a peer can provide. An "overlay network" is a layer of virtual network topology on top of a physical network topology, as shown by Figure 2.10. The main functionality offered by an overlay network is efficient message routing. It is necessary to carry out a detailed discussion about P2P overlay networks

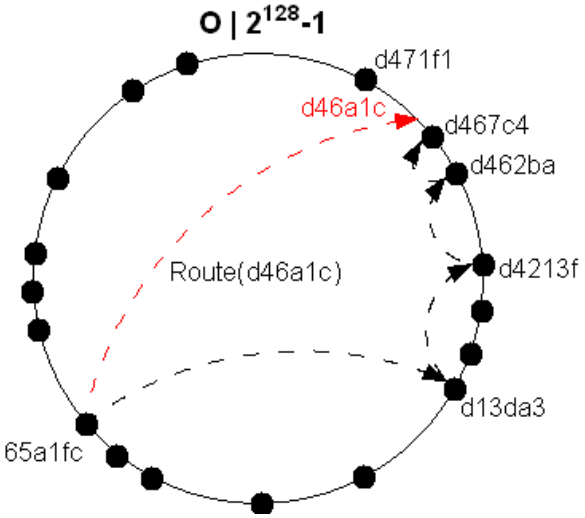

Figure 2.11: The Pastry overlay network

in this section, because both the Mediator framework (Chapter 4) and Deadline-Driven Auctions (Chapter 6) are deployed on P2P overlay networks.

Take the Pastry overlay network as an example. Peers are organized into a logical ring depicted by Figure 2.11. Each peer node in the ring has a unique, uniform random identifier (i.e. nodeId) in a circular 128-bit identifier space. When presented with a message and a numeric 128-bit key, a Pastry node efficiently routes the message to the node with a nodeId that is numerically closest to the key, among all currently live Pastry nodes. For example, in Figure 2.11, when peer node $65a1fc$ sends out a message with key $d46a1c$, the message is routed (i.e. forwarded) by node $d13da3$, $d4213f$, $d46sba$, and is finally received by node $d467c4$, because currently this nodeId is numerically closest to the key.

Overlay networks provide the following capabilities:

Self-organization A peer only requires minimum configuration and knowledge about other peers in order to join an overlay network. For example, in Pastry each peer only needs to know one existing peer in the overlay network, and then the peer can join the

overlay network by first generating a random nodeId for itself using a hashing function like SHA-1 [3], and then sending a series of bootstrapping requests to the known peer.

Automatic load-balancing Because both nodeIds and message keys are randomly assigned and uniformly distributed, without requiring any global coordination, this results in a good first-order balance of query load among all the peer nodes, as well as network load over the underlying Internet.

Fault-tolerance An overlay network is fairly resilient to peer node arrivals, departures and failures. For example, each Pastry node keeps track of l immediate neighbors on both sides in the nodeId space, which is called the node's leaf set. Only when more than $l/2$ adjacent nodes in a leaf set leave or fail simultaneously is a Pastry ring broken. Otherwise, any node exceptions can be recovered by exchanging $O(log_{2^b} N)$ messages.

Efficient and scalable information dissemination Structured network locality properties ensure that information can be efficiently disseminated even in large scale P2P systems. For example, in Pastry given a node population N, each peer node maintains a local routing table of $log_{2^b} N * (2^b - 1) + l$ entries (b and l are configurable, by default $b = 4, l = 16$), and a message can be delivered within $log_{2^b} N$ hops.

Mapping application objects to peer nodes Besides messages, any application-specific objects may be identified by a key (objectId) as well. In this way, an object can be mapped to an "owner" node, whose nodeId is numerically closest to the objectId. Insertion of an object is handled as the routing of a message, where the objectId is used as the object's destination. Accessing an object involves retrieving the object from its owner node, where the objectId serves as a lookup key.

Enhanced availability and persistence Instead of assigning an object to a single owner, replicas of the same object can be stored at multiple owners. In this case, even though

some of the owners are temporarily disconnected, the object is still available at many other places.

In summary, structured P2P overlay networks provide a generic, scalable and reliable substrate for building P2P applications. With their help, efficient request routing, deterministic object location and load-balancing can be achieved conveniently in an application-independent manner. For example, Pastry has been applied to a Scribe P2P group communication and event notification infrastructure [38], PAST P2P archival storage middleware [144], ePost cooperative Email service [123], SplitStream high-bandwidth content distribution infrastructure, and several other research projects.

2.5 Advantages and Challenges of P2P MMOGs

2.5.1 Advantages

Compared to the conventional C/S architecture, a major attraction of the P2P architecture is that it provides a scalable and low cost way to build MMOGs. P2P systems are becoming increasingly popular, as they are inherently good at sharing heterogeneous computing resources in a distributed fashion while keeping costs down. Recently, as the computing power available on personal machines has grown dramatically along with cheap network bandwidth, it has become possible to migrate a major part of a game server's functionalities to resource-rich client machines, so as to support a MMOG using a pooling of game participants' computing assets, including CPU cycles, memory, storage capacity and network bandwidth. Technically speaking, the P2P architecture has several advantages over a C/S architecture:

- ***Reliability*** A P2P MMOG distributes the functionalities of a conventional game server to many game participants (a.k.a. super-peers), each carrying out a bit of the total workload. A system like the Mediator framework (Chapter 4), that is

designed for resilience, enhances the dependability of a super-peer using a number of super-peer backups. As a result, the failure of a super-peer may only affect the gaming experience of a limited amount of players in a short period of time, before a super-peer backup is activated to take over the responsibilities of its predecessor.

- ***Scalability*** A P2P architecture is more scalable than a C/S architecture, as more players will also bring more computing resources to a P2P application like a MMOG. Currently, a gaming computer is often equipped with a fairly powerful CPU, gigabytes of memories, and adequate network bandwidth. Therefore, it is feasible for a participant machine to carry out additional administrative or computational tasks for the collective welfare besides its normal support for playing. Furthermore, an incentive mechanism (Section 3.7) can help to convince application participants to contribute available computing resources, as well as to maintain a sufficient level of reciprocity.

- ***Redundancy*** Because a P2P MMOG relies on computing resources that are available at game participant machines, it is inherently immune to the redundancy problem. A P2P MMOG only needs to rely on a centralised game server for hard to diversify services that consume few resources like account authentication, so that accurate estimation about the server's hardware configuration is no longer an important issue.

- ***Cost*** From a commercial perspective, a P2P MMOG offers the possibility to deploy a MMOG without subscription fees without the incredible investment required by previous architectures. It also offers business opportunities to the game industry, because it is more flexible and profitable for enterprises to market their game software without providing dedicated hardware support. In other words, the business risk for launching a MMOG project can be significantly reduced.

- ***Sociodynamics*** A P2P MMOG is especially good at load-balancing, because players roam in the game world together with their computing resources. In a C/S architecture, a region server may be overloaded when the flocking problem occurs.

However, from a P2P system's point of view, even a crowded game region is still self-sufficient in terms of computing resource.

2.5.2 Challenges

A P2P MMOG is more complex than any existing P2P applications mentioned in Section 2.4.2, and the feasibility of adapting MMOGs to the P2P architecture is still in doubt due to the following challenges:

Game Interactivity One of the most important features of a MMOG is real-time interaction. A MMOG is vulnerable to latency challenges (Section 2.2.4), as Quality of Service is crucial to a player's gaming experience, which in turn determines a MMOG's business success. In other P2P applications, for example file sharing, one may only attach importance to an application's average download speed. But in a MMOG, the delay for transmitting every game event needs to be kept small enough nearly all the time. However in P2P networks it is hard to guarantee the performance of all the connections among game peers.

Minimum Number of Users For many P2P applications, a minimum number of users is critical to their success, because the more users that participate, the more efficient the application will be. When it comes to a P2P MMOG, it is important for game designers to ensure a game's availability. On the one hand, any node failure must not influence the process of the game. On the other hand, even if the online population is small, the game world should still be able to evolve normally and efficiently.

Asymmetric Network Bandwidth Because a peer node of a P2P network represents both client and server parts of an application, it will probably utilize as much bandwidth going in as going out. However, this is not how Internet service providers are configured

to work. In fact, most Internet users experience asymmetric network bandwidth, which in turn requires that the communication overhead imposed on each player machine must be regulated below a specific threshold independently of the number of concurrent players.

General Obstacles P2P networks are confronted with some inherent obstacles that need to be taken into consideration. For example, one prevalent problem on the Internet is barriers like firewalls and proxy servers that prevent direct communications among peers. In order to circumnavigate such barriers, techniques like HTTP tunneling are indispensable. Moreover, since no node is more important than any other in pure P2P networks, many of them may need to be connected to each other. Obviously, such a pure P2P architecture may not scale well in that when the quantity of peers increases, the augmentation of connections between peers may mount up egregiously. So, only carefully designed P2P architectures, which may need to be hybrid, may prove practical in the end for implementing a P2P MMOG if such challenges are to be overcome.

Security As a game server's responsibilities are carried out by common player machines, a P2P MMOG is less easy to secure than a centralized C/S architecture. In other words, a downside risk of adopting a P2P approach to a MMOG is that it may make it too hard to stop cheating.

A P2P MMOG must overcome all the problems listed above, and even more. Chapter 3 systematically discusses the essential design issues presented by P2P MMOGs, together with a survey and classification of related research, explaining what problems have been addressed in the literature, how they were dealt with, and what their relationship is to this thesis.

— We can be knowledgeable with other men's knowledge, but we cannot be wise with other men's wisdom.

Michel de Montaigne

Chapter 3

Design Issues for P2P MMOGs

3.1 Introduction

To adapt MMOGs from conventional C/S architectures to P2P architectures is a challenging and active research area. This chapter articulates a broadly comprehensive set of six key issues for the design of P2P MMOGs, namely interest management (Section 3.2), game event dissemination (Section 3.3), NPC task sharing (Section 3.4), game state persistency (Section 3.5), cheating mitigation (Section 3.6), and incentive mechanisms (Section 3.7). Design alternatives for each issue are systematically compared, the relationships between design decisions are discussed, and areas for further research are identified. Also, several representative P2P MMOG infrastructures are classified and evaluated against the design issues (Section 3.8).

3.2 Interest-Management

In a game with a conventional Client/Server architecture, every player communicates with a central game server, which is then responsible for broadcasting game state changes

to all players in the same game zone, as discussed in Section 2.3.1. In this case, an immediate requirement on a P2P MMOG is how to maintain this consistent, shared sense of virtual space among large numbers of players without the server's support. To ask each player to maintain a full copy of the game state and all players to broadcast updates to all other players in the game is not a viable solution, because this approach does not scale well. As the number of players increases, the messages sent over the network increase exponentially. In the literature, sophisticated Interest-Management (IM) techniques have been proposed to overcome this problem [163, 168, 88, 140].

IM is a classical research topic in VE, and was initially addressed by Morse et al. in the early 1990s [119]. The concept of IM originates from two observations. A single player does not need to know about what is happening in the virtual game world as long as it does not affect the player; and a player's avatar only has limited movement speed and sensing capability. So, a player's view of the game world can be limited to a comparatively static Area of Interest (AOI), and the player only needs to subscribe to game events that occur within its AOI. IM is the process of determining which information is relevant to each player [119].

Existing IM schemes can be classified into at least three broad categories: a spatial model, a region-based publish/subscribe model, and a hybrid communication model, and these are described next.

3.2.1 Spatial Model

The spatial model uses the properties of space as the basis for mediating interaction [27]. This model is also referred to as the "aura-nimbus" model [31, 125] because of its key abstractions: aura and nimbus. The "aura" means the area that bounds the presence of an object in space, while the "nimbus" means the mutual awareness levels between two objects. In other words, object A is only able to interact physically with object B when their auras intersect with each other, but A is aware of B when it is in B's nimbus.

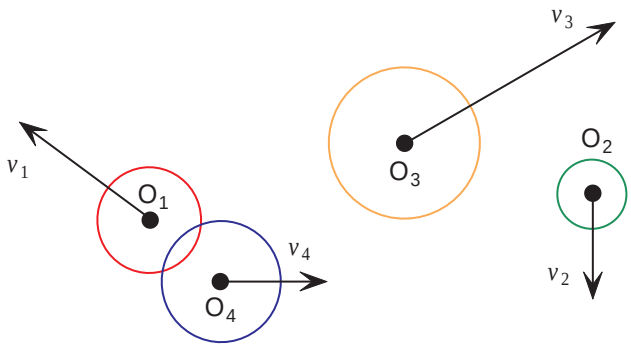

Figure 3.1: Representation of moving objects using the aura-nimbus model

So, every object should establish communications with other objects that fall within its nimbus, in order to prepare for potential interactions.

Usually, the aura can be modeled by a solid point that represents an object's position in a virtual world, and the nimbus can be modeled by a fixed-size circle around the aura that represents the object's AOI. Figure 3.1 illustrates four moving objects in a game world using such notations. Furthermore, each object's moving direction and velocity are also indicated by arrows. In a MMOG, different objects may have different sensing and moving capabilities. For example a player having a telescope can see further, and a player riding a horse can move faster. These differences are reflected by the range of a player's AOI and the length of a player's speed arrow.

The advantage of a spatial model is that it allows fine-grained IM in which only necessary messages are transmitted among relevant peers [31]. However, a significant drawback is that it requires all objects to exchange positional update information in order to identify when AOI collisions occur. For example, in [120] all players must share their "frame of reference" to know the location of each other, so that each player is able to discover the "active entities" within its AOI. The frequency of these updates must be sufficient to ensure that AOI collisions can be determined in a timely fashion, which may lead to considerable communication overhead and undermine the scalability of the system [125].

To mitigate the limitation of a pure spatial model, a Voronoi diagram can be employed to

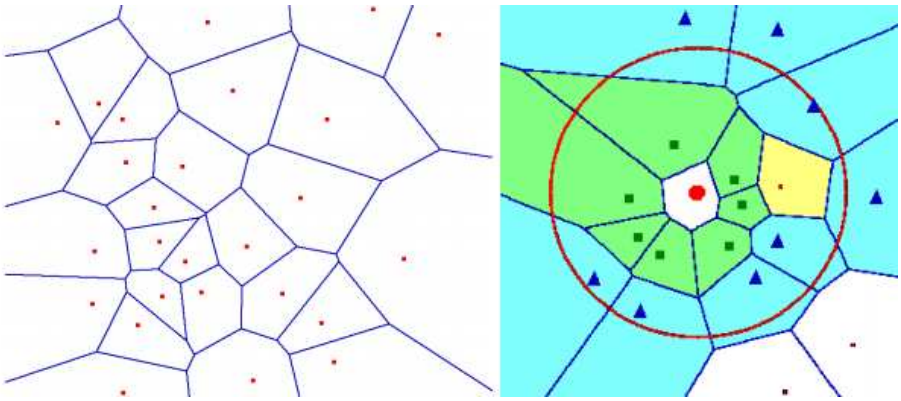

Figure 3.2: Neighbour discovery in a Voronoi diagram [88]

help a player find its neighbouring players in a virtual world (i.e. neighbour discovery) [34, 88, 86]. Given n points on a plane (each point called a "site"), a Voronoi diagram is constructed by partitioning the plane into n non-overlapping regions that contain exactly one site in each region. A region contains all the points closest to the region's site than to any other site. The entire plane is therefore divided into arbitrary sizes in a deterministic way [88]. The left part of Figure 3.2 illustrates such a Voronoi diagram, which can be used to find the k nearest neighbours of a specific site.

In the diagram, a player's k nearest neighbours are further classified into Enclosing and Boundary neighbours. Enclosing neighbours are defined as regions that share a common edge with the player's own region. Boundary neighbours are defined as regions that overlap with the player's AOI. The right part of Figure 3.2 shows the detailed classification of neighbours, where Enclosing and Boundary neighbours are marked by squares and triangles respectively (an Enclosing neighbour can also be a Boundary neighbour).

In a P2P MMOG implementation, each peer can be required to construct and maintain a Voronoi diagram by itself, based on the spatial coordinates of neighbours. A peer only needs to keep network connections with its current neighbours, because potential neighbours can be discovered with the help of its Boundary neighbours. In other words, each peer serves as the "watchman" for one another in discovering approaching neighbours. When a peer moves, position updates are sent to all neighbours recorded in the Voronoi

diagram. If the receiver is a Boundary neighbour, an overlap-check is performed. The receiver checks if the mover, with its new AOI, would enter into any of its Enclosing neighbours' Voronoi regions. The receiver only notifies the mover if a new overlap occurs. This allows the moving peer to become aware of potentially visible neighbours outside its AOI.

The Voronoi approach is able to reduce the communication overhead incurred by a pure spatial model, because each peer only needs to maintain network connections and exchange messages with a limited number of neighbours. However, the approach is not ideal in that:

- A Voronoi diagram is vulnerable to the "circular line-up" problem, which is the worst case when a peer has $n - 1$ neighbours in a diagram of n sites.
- The communication overhead is not minimum, because a peer still needs to receive and process messages outside its AOI.
- While the communication overhead is reduced, the computation overhead is increased as every peer is required to construct and maintain a Voronoi diagram locally.
- A gateway server is needed for boot-strapping purposes, through which a joining peer finds its acceptor region, i.e. the region that contains the joiner's initial coordinates.

3.2.2 Region-based Publish/Subscribe Model

In contrast with the spatial model, the region-based publish/subscribe model proposes to support a P2P MMOG using coarse-grained IM. In this model, the virtual game world is partitioned into well defined static regions, and the recipient of a message is limited to only interested participants that reside within the same, or neighbouring, region as

the sender [125]. In this case, IM can be abstracted using a publish/subscribe model, in which publishers are objects that produce events, and subscribers are objects that consume events [31]. An object, e.g. a player's avatar, can be both a publisher and subscriber.

The main responsibility of a region-based IM mechanism is to determine the regions that intersect a player's AOI, and to form the area-of-subscription from the union of the intersected regions. Usually, each region is implemented as an individual channel for publishing gaming events. So, if a player's AOI intersects n regions, the player needs to subscribe to n different channels at the same time in order to receive all relevant events.

The region-based publish/subscribe model offers many advantages. Firstly, it is straightforward to design and implement, because to compute a player's area-of-subscription is often simpler and cheaper than to compute AOI collisions in a aura-nimbus model. Secondly, a region channel maps nicely onto a multicast group, hence both publishers and subscribers may send and receive gaming events efficiently. Last but not least, because all regions are predefined and released with the game software as a part of the game world's description, players are able to carry out local IM without knowing in advance the position and moving states of other players.

However, this model also suffers from two major drawbacks. On the one hand, the quality of region-based IM is highly dependent on the shape and size (i.e. granularity) of regions. With reference to the shape, no definitive solution is given in the literature, and both hexagonal [68, 168] and quadrant zoning [118, 165] have been equally employed in the related work. Recently, it appears that hexagonal zoning has become preferred by game developers for the following reasons:

- Hexagons have uniform orientation and uniform adjacency.
- In hexagonal zoning, a participant moving from one cell to another joins and leaves the same number of cells.

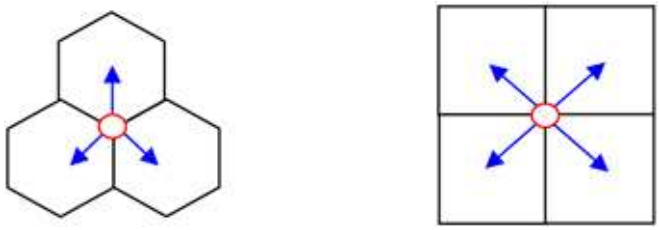

Figure 3.3: Worst cases for a subscriber in hexagonal zoning and quadrant zoning

- Because an AOI is usually defined by a radius, hexagonal zoning approximates more to a circle than quadrant zoning.
- In the worst case of hexagonal zoning, a single player needs to subscribe to three regions at the same time, whereas this number is up to four in quadrant zoning, as shown in Figure 3.3.

Regarding the size of a region, it has been pointed out that a region must be of sufficient size as to ensure objects are able to disseminate messages in one region before entering another region [125]. However, the granularity cannot be too large, otherwise a player's machine might be overloaded by excessive irrelevant messages outside its AOI. It has been suggested that an appropriate granularity should result in approximately one third (in hexagonal zoning) of the communication load a specific game engine generates under a C/S architecture, so that the same level of load would be reached while a player is simultaneously interested in three hexagons [68] in a P2P MMOG. Nevertheless, up to the present no easy way has been developed to conform to such a principle.

On the other hand, region-based IM does not work well when objects are not evenly distributed among the regions. Specifically, if too many objects gather in the same region as the player, or even in neighbouring regions, the player may still receive too many state updates from invisible objects outside its AOI. To mitigate this limitation, mechanisms have been proposed to organize gaming event multicast paths based on the proximity of peers in the virtual world. These improved mechanisms organize peers in "clumped" regions into dynamically balanced N-Trees [163], which enable a peer to discover other

peers that are within a specific proximity. Therefore, a peer can propagate a "scoped event" only to those close by peers, ignoring other further peers although they are in the same region. Of course, the construction and maintenance of a N-Tree will introduce extra communication and computation overheads, as well as a series of reliability issues as the tradeoff.

3.2.3 Hybrid Communication Model

As indicated by its name, a hybrid communication model is a mixture of the spatial model and region-based model that have been discussed previously. The MOPAR [168] and Meta-Model [140] infrastructures are representatives of recent work on hybrid IM for P2P MMOGs, which combine a structured P2P overlay network and a super-peer network with the spatial model.

First of all, a hybrid communication model partitions a game world into multiple regions, and identifies each region using a unique regionId in a structured P2P overlay network, as discussed in Section 2.10. Then, a super-peer is selected to be responsible for the IM task in each region according to predefined criteria. A super-peer refers to a peer in a P2P network that operates both as a server to a set of clients, and as an equal in a network of super-peers [166]. Super-peer selection [117] is a complex issue that is beyond the scope of this Chapter. Once a super-peer is selected for a region, its peerId is stored by another peer in the overlay network, whose peerId is numerically closest to the regionId.

When a player wants to join a specific region, the player can find out the current super-peer working for that region by routing a query message to the regionId through the overlay network. Next, the player establishes communication with the super-peer, and updates the super-peer when its moving speed or direction change. In this way, the super-peer obtains a global view of the region, anticipates every peer's position in the near future, and analyzes AOI collision events using an algorithm such as [151]. Players, whose AOIs are about to intersect, will be notified by the super-peer to establish direct

P2P connections with each other, so they can prepare for potential real-time interactions. So, from a common player's point of view, an IM service is provided in each region using the spatial model.

The hybrid communication model takes advantages of both spatial and region-based models. On the one hand, it facilitates fine-grained IM in each region and reduces more communication overheads for common players than in a pure region-based model. On the other hand, it is relatively simpler to implement and more efficient than a pure spatial model. However, it may impose high computation and communication workloads on a super-peer for a crowded region, so dynamic zoning techniques are needed as a remedy to distribute the total workload to a set of super-peers, each taking charge of a sub-region. Furthermore, a super-peer is potentially a single failure point in a region, so fault-tolerant mechanisms are also needed to provide suitable super-peer backups in order to achieve adequate robustness for the system.

3.2.4 IM Discussion

Of the three IM approaches outlined above, the spatial model is the most fine-grained, and as we shall see in the next section, is a prerequisite for disseminating game events using unicast. However, due to the lack of centralised control, the communication overhead for a player to establish a global view of the game world may be high. This can be mitigated with a hybrid model that combines the advantages of both the aura-nimbus and region-based models. As hybrid models use super-peer networks, they must address super-peer selection, load-balancing and fault-tolerance issues.

3.3 Game Event Dissemination

While IM focuses on finding out what information is relevant to each player, game event dissemination is concerned with how relevant information is actually delivered to the

players. In fact, the choice of a game event dissemination approach is largely determined by the underlying IM mechanisms used in a P2P MMOG.

3.3.1 Unicast or Multicast

A spatial model supports fine-grained IM, which explicitly tells a small set of objects that a player may interact with them shortly. Therefore, the player just needs to establish direct P2P connections with these objects, and gaming events are exchanged through unicast communications, e.g. in [120, 34, 88, 86]. Similarly, because super-peers in a hybrid model also provide fine-grained IM services, unicast applies to [168] and [140] as well.

However, a coarse-grained IM scheme only tells a player of some regions to which he should subscribe. In these regions there might be a large number of players, so unicast via direct P2P connections is infeasible. Instead, each region is represented by a multicast group, which offers a single medium for any region participant to publish gaming events in, and enables the events to be received by all the other region participants. In this circumstance, multicast technology becomes crucial for game event dissemination.

3.3.2 Application-Level Multicast

Traditionally, IP multicast [56] was proposed as a mechanism for sending messages to a group of recipients efficiently [68]. However, as IP multicast has a number of technological, practical, and business obstacles [58], it is not widely available on the Internet. As an alternative Application-Level Multicast (ALM) has been proposed to support similar functionality, but as an application service instead of a network service [63].

Structured P2P overlays provide good communication infrastructures for building ALMs. For example, Bayeux [176] uses Tapestry [175], CAN Multicast [135] uses CAN [134], and both Borg [174] and Scribe [38] use Pastry [143]. All the ALM systems mentioned

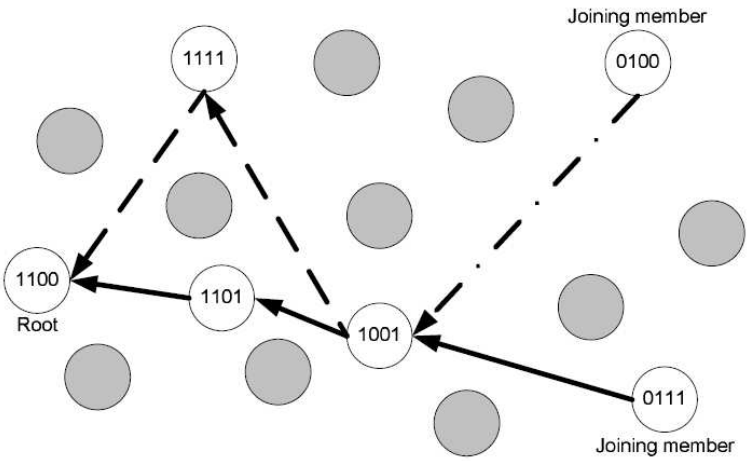

Figure 3.4: Reverse path forwarding and path maintenance in Scribe [38]

above are tree-based, which means that participants are organized into a hierarchical multicast tree. Because the Mediator framework and MAMBO technology described in Chapters 4 and 5 are implemented using Scribe, this section takes Scribe as an example to explain how a multicast tree is created.

Scribe uses a Pastry key as the groupId for each multicast group, and the peer node with the nodeId that is numerically closest to the groupId serves as the rendezvous point of the group (a.k.a. the root). A multicast tree associated with the group is formed by the union of the Pastry routes from each group member to the root. Multicast messages are delivered from the root to the members using reverse path forwarding.

In Figure 3.4, suppose that peer 1100 is chosen to be the root for a multicast group, which peer 0111 wants to join. In this case, peer 0111 sends out a joining request message towards the root using the Pastry routing algorithm. The message is successively forwarded by peer 1001 and 1101 (a.k.a. the forwarders) on the Pastry ring, and finally arrives at the destination, as shown by the solid arrows. Later on, when peer 0100 subscribes to the same multicast group, its joining request might also be routed via peer 1001 on its way towards the root, as shown by the dot-dashed arrow. Because at this time peer 1001 is already a forwarder in the multicast tree, it can complete the subscription process immediately on behalf of the root.

Effective construction of a Scribe multicast tree is guaranteed by two locality properties of the Pastry routing algorithm. The *short routes* property ensures that in each step a message is routed to the nearest node in terms of network proximity. As a result, the *route convergence* property ensures that the routes for messages sent towards the same destination will converge quickly. In other words, there is a high probability for joining requests from different peers, e.g. 0100 and 0111, to converge at the same forwarder peer like 1001. Therefore, group member management in Scribe can be efficient, as adding a member to a group merely involves limited routing activities before a joining request reaches any existing member of the tree.

At publication time, a participant firstly sends the contents to be published to the root, which in turn delivers the contents to all the closest forwarders, such as peer 1101. Then, using reverse path forwarding, the forwarders continue to relay the contents until they reach the final subscribers, such as peer 0100 and 0111. It is important to notice that though the intermediary forwarders may not be interested in the contents that are published, they are still recruited as members of a multicast group. So, given a Scribe multicast tree, it is only guaranteed that all the leaf peers are subscribers, whereas the internal peers are either subscribers or forwarders. Furthermore, if there are multiple multicast groups in an application, the same peer may be employed to work as a forwarder for more than one group.

Path maintenance in Scribe is efficient, too. For example, in Figure 3.4, when peer 1101 leaves the system, peer 1001 will detect the exception and initiate a new joining process. As a result, peer 1001 reattaches itself to another forwarder 1111 as shown by the dashed arrows, but other subscribers that depend on 1001 are not affected by this change at all.

Considering the advantages and convenience provided by an ALM system, several researchers [118, 79, 57, 91] propose building a P2P MMOG using a region-based IM scheme and disseminating gaming events in each region with Scribe. Mercury [136] resorts to content-based publish/subscribe rather than a simple channel-based approach. However, the key idea is similar in that Mercury also organizes game participants into

a P2P overlay for a range-query, and supports event dissemination using ALM. In addition, the architectures proposed in [165, 142, 46, 109] fall into the same category as well, where their main novelties are distinct group member management and multicast tree construction algorithms that place emphasis on the load-balancing capability, robustness and scalability of an ALM system.

3.3.3 Problems with ALM

A significant problem with disseminating gaming events using an ALM system is the potential latency issue. In unicast based P2P MMOGs, players exchange real-time gaming events with other relevant players directly, where the only possible cause of latency is the network delay between the two players. However, in ALM based P2P MMOGs, in order to publish a gaming event, a player first needs to send the event to the root of a multicast tree, which then takes additional multiple hops to deliver the event to target receivers. This process incurs unnecessary end-to-end delay, especially when the size of a multicast group is large and an event has to travel through a deep multicast tree. Smart ALM systems, e.g. Scribe, try to address this by carefully selecting forwarders in order to reduce the latency as much as possible. However, simulations performed on several network topology models show that the average distance traveled by a message is between 1.59 and 2.2 times the actual distance between the source and destination in the underlying Internet [38], which means that ALM is generally slower than unicast.

To cope with this problem, it has been suggested that a multicast tree can be constructed according to the proximity of peers in the game world instead of the proximity of peers on the network [163, 72, 146]. In this way, players in the vicinity are employed as the most immediate forwarders, hence multicast messages get sent to closeby peers faster, while peers that are further away receive them more slowly. This approach attempts to provide a better gaming experience by exploiting the tolerance of distant players in a game world for weak synchronizations.

A general expedite event dissemination mechanism for P2P MMOGs [17] has also been proposed in the literature. No matter what environmental setup is used, the mechanism is able to reduce the overall time that a multicast tree takes to disseminate an event. The key idea is to utilize better a forwarder's "idle period", which is the time slot between a forwarder's completion of a relaying task and the generation of the next gaming event.

3.3.4 Event Dissemination Discussion

Disseminating game events with ALM will typically induce a larger communication latency than with unicast. However, efficiency is also a consideration for unicast as a player may not have enough bandwidth to send every game event to large numbers of recipients. Hence a P2P MMOG should employ either a fine-grained IM mechanism that enables a player to unicast game events only to necessary recipients, or a specially designed ALM mechanism that is able to exploit distant players' tolerance of weak synchronization.

3.4 NPC Task Sharing

Besides player controlled avatars, there are also considerable numbers of NPCs in a MMOG, which are crucial elements in game scenarios as discussed in Section 2.2.2. Traditionally, the NPCs are hosted by a central game server, consuming significant processing power and network bandwidth. So, one of the prerequisites for realizing a P2P MMOG without a server's support is to provide a mechanism that hosts such NPCs using computing resources that are available on common game participant machines. From this perspective, the distributed hosting of NPCs can be viewed as a task sharing problem in a heterogeneous computing environment.

A NPC task has the following characteristics. Firstly, it consumes processing power, as each NPC is associated with a "think function" [29], which is the AI program that

determines the NPC's actions. For example, a monster NPC may determine its move by examining the surrounding terrain and the position of nearby players. Secondly, it is interactive, as a NPC needs to communicate with or behaviourally respond to other players. For example, a monster NPC may engage in combat with player avatars, which would result in the exchange of real-time gaming events. Last but not least, it is indivisible, as a NPC can usually only be efficiently hosted by a single computer.

According to the characteristics listed above, the responsibilities of a NPC task sharing mechanism include the selection of a host computer that is able to provide an adequate computing resource, the migration of related think function and state information for a NPC, and a NPC location service for other players to get into contact with a NPC's host. From this point of view, a NPC task sharing mechanism is also referred to as a NPC host allocation mechanism. Many different NPC host allocation schemes have been proposed in the literature, which can be classified into at least two broad categories - static region based approaches and dynamic virtual distance based approaches.

3.4.1 Static Region Based Approaches

Static region based approaches [118, 91] partition a virtual game world into multiple regions, and assign each region a super-peer, which works as an authoritative server and hosts all the NPC objects within the region.

In [118], a live peer whose peerId is the closest to the regionId is selected as the "coordinator" for that region. It receives and processes all gaming events in the region, as well as multicasting state updates to region participants. Because a region coordinator is selected according to the Id locality in a structured P2P overlay network, it is unlikely that the coordinator is a member of its own region. In other words, a coordinator is responsible for a region regardless of its actual position in the game world, and the NPC host is only changed when the coordinator leaves the application, or another peer shows up with a peerId that is closer to the regionId.

The super-peer selection policy is slightly different in [91], which requires a "zone owner" to be a member of its own zone. If the zone is empty, the first peer that joins the zone becomes the zone owner. Otherwise, a successor is appointed by the previous zone owner when it leaves the zone or the application. Similar to [118], a structured P2P overlay network is employed as the rendezvous point for an application, and the peer whose peerId is the closest to a zoneId serves as the register for that zone, telling a newly joined peer whether it is the first member of the zone, or whether there is already a working zone owner that the peer should contact.

Static region based approaches have several significant drawbacks. Firstly, they might incur excessive computation and communication workloads on a super-peer, e.g. a region coordinator or zone owner. Secondly, the super-peer selection mechanisms are overly simple, especially when random peerIds and regionIds are used. A P2P MMOG is a heterogeneous computing environment, so it is impractical to assume that peers' hardware configurations are similar and resource availabilities are equally adequate. Last but not least, these approaches cannot guarantee to fulfil the real-time requirements of PC-vs.-NPC interactions. Some region participants may suffer from slow connections to a super-peer, even though the super-peer is able to provide enough computing resource.

3.4.2 Dynamic Virtual Distance Based Approaches

In contrast with static region based approaches, which are somewhat centralized, dynamic virtual distance based approaches completely distribute NPC objects to all game participants. The key idea in these approaches is to allocate a NPC object to the player, whose avatar is closest to the NPC in a virtual game world. The rationale is that a player that is closest to a NPC is most likely to interact with the NPC. So, if the player is hosting the NPC by itself, it removes the need for the player to communicate with a potentially remote third party. It has even been suggested that the virtual distance based approach is optimal for minimizing interactive latency and communication overhead [29].

Colyseus [29] has demonstrated the feasibility of applying virtual distance based task sharing in Quake II, a well-know multiplayer first-person shooter game. The game object manager of Colyseus allocates mutable objects, e.g. NPC characters, doors and weapon items, to the closest players, and supports a object locator service using either a traditional randomized Distributed Hash Table (DHT) [150, 143], or a range-queriable DHT [136]. However, Colyseus does not provide technical details about how its "placement strategy" determines which player's avatar is closest to a mutable object in terms of virtual distance.

A concrete algorithm for virtual distance calculation is given in AtoZ [167]. AtoZ allocates each player avatar a "priority field", which is analogous to Mahalanobis distance in the domain of quadratic discriminant analysis [21] to decide which avatar can access the shared object in a shortest time. This approach is dynamic in that the priority field is decided dynamically, considering the relationship between the associated avatars in terms of their position, velocity, acceleration and orientation. The weakness of AtoZ is that sometimes the host for a shared object is ambiguous in a "dead zone", which is the border area between two players' priority fields. Furthermore, the workability of the algorithm has only been demonstrated with an air-hockey game that involves two players and a single hockey object, whereas its effectiveness and efficiency in a multi-user and multi-object environment have not been evaluated.

The Voronoi diagram discussed in Section 3.2.1 seems inherently suitable for virtual distance based task sharing. As shown in Figure 3.2, a Voronoi diagram partitions a game world into multiple non-overlapping regions that contain exactly one player avatar in each region. In this case, it is natural for each player to host the NPC objects within its own Voronoi cell [86, 34]. In [86], a player has two separate roles: "arbitrator" and "peer". An arbitrator is in charge of all NPC objects in a Voronoi cell, and has the authority to decide how game states should change according to event messages and game logic. A peer is a node that generates events and displays the results of processed events. All players start as both peers and arbitrators where each peer submits events to its arbitrator part for processing. However, when players crowd together and become

overloaded due to increased connections and messages, a more capable player is selected as the only arbitrator responsible for a group of peers. This elevated super-peer is called an "aggregator".

Compared to static region based approaches, dynamic virtual distance based approaches are better at utilizing the computing resources of more player machines, eliminating interactive latency for some of the players (i.e. NPC owners), and reducing the communication overhead. However, they also have the following disadvantages:

- While eliminating latencies for NPC hosts, the approaches may lead to high latencies for non-host players. Of course, it is likely that a player closest to a NPC will interact with the NPC, but it does not mean that other nearby players will not. Contrarily, it is quite usual for a group of players to attack the same monster in a MMOG. In this case, all non-host players need to communicate with the host, and it is not guaranteed that the latency for each player is equally small.

- The computation of accurate host allocation can be expensive, and because a majority of the players in a MMOG are constantly moving, switches of NPC host may be frequent. Therefore, the overall computation and communication overhead may be still high.

- These approaches cannot account for the spawning/respawning mechanisms that create all the NPC objects in the first place. In this case, a centralized game server that initially controls all the NPC objects is still needed. For example, in [86] NPCs are managed collaboratively by both servers and clients. To avoid ambiguities in NPC host allocation caused by player movements, as well as overloading of busy arbitrators, the game server exerts an important role.

- Cheating may become easier for unscrupulous players who might abuse their hosting of NPC objects to their own advantage. When the think function and game logic for a monster that combats with a player are maintained by the same player, the rules of the game may be subverted. For example, an unscrupulous player may

tamper with the game logic so as to kill a monster without effort. Even worse, because no third party is required in a local interaction, it is rather hard to detect such a breach.

3.4.3 NPC Task Sharing Discussion

Currently, virtual distance based approaches are more widely used for NPC task sharing because they minimise the communication latency for NPC hosts. In this thesis, a novel NPC task sharing mechanism *DDA* is proposed, which further optimises overall communication latency when an NPC interacts with multiple players. Potentially P2P MMOGs could utilise both strategies flexibly in different game scenarios.

3.5 Game State Persistency

A MMOG is also referred to as a Persistent World, because the game world is always available to the users and game plots evolve even while some of the players are not playing their characters online, as discussed in Section 2.2.1. In this case, a MMOG must store all players' profiles and inventories, as well as any changes they have affected on the world, between login sessions. When a player comes back to the game, the player retrieves its previous state information and continues to play. Usually, a MMOG operates for many years, and during this time substantial amounts of data may need to be kept.

Conventionally, such game data is preserved by centralized databases in game servers. However, once a game server is down, important game data might be lost. For example, Blizzard's World of Warcraft game server crashed on 21 Jan 2006. This eventually led to compensation for thousands of players, whose inventories were damaged in the accident [112]. As a result considerable research effort has been put into the development of reliable, scalable, and high performance persistency solutions for MMOGs [112, 173]. This issue becomes especially challenging in the context of a P2P MMOG, where centralized management is not available.

3.5.1 Distributed Storage Infrastructures

Several distributed storage infrastructures have been proposed in the literature, which may facilitate game state persistency in P2P MMOGs. For example, the OceanStore project [104] provides a global persistent data store utility designed to scale to billions of users. It supports consistent, highly–available and durable storage atop an infrastructure comprised of untrusted hosts. OceanStore assumes that any host may crash without warning or leak information to third parties. So, sophisticated redundancy and cryptographic techniques are introduced to protect the privacy, integrity and availability of the information being stored.

Furthermore, large-scale persistent storage services have also been built upon structured P2P overlay networks, e.g. PAST [144] that uses Pastry. Any host connected to the Internet can act as a PAST node that contributes its local storage capacity to a distributed application. PAST assigns each file that is stored in the system a unique fileId, and replicates the file at k PAST nodes whose nodeIds are numerically closest to the fileId. Operations like $Insert$, $Lookup$ and $Reclaim$ are supported by the underlying Pastry routing algorithm. Compared to OceanStore, PAST is preferred by more P2P MMOG systems [79, 23, 29, 91]. A significant reason is that most of them already employ Scribe for game event dissemination. Scribe is an ALM service that is also built upon Pastry, as described in Section 3.3, so it is fairly simple and convenient to support Scribe and PAST at the same time.

3.5.2 Further Considerations

Though the distributed storage infrastructures discussed above provide many advantages, their suitability to be applied to a P2P MMOG directly is still in doubt. One of the considerations is the efficiency for reading and writing data through such infrastructures. It has been already pointed out that a MMOG may generate event updates frequently, but it takes a long time to modify the data using an overlay network. In this case, there will be

a severe delay when a player wants to retrieve the data that is still being modified. So, it has been proposed that data holder super-peers be used to provide a caching mechanism, so that real-time event updates take effect immediately, whereas a distributed storage infrastructure is only used as a slow medium for backup purposes [91].

Moreover, availability is another significant consideration. It has also been argued that data should always be available for retrieval, since players would not be satisfied if their characters were inaccessible because the persons that were storing their characters left the game [23]. This related work addresses the problem of separating storage needs into two categories: permanent and ephemeral data.

Ephemeral data makes up the bulk of data in a MMOG. For example, a player drops a sword in the forest. The sword should be stored as ephemeral data because if the player leaves the game, other players can still see the sword on the ground for some time. However, it does not matter much if most of this data is lost, because its loss would not usually affect the overall game. On the other hand, character data and inventories are examples of permanent data. Data which is ephemeral can be simply stored using a distributed storage infrastructure, indexed by its geographical area in the virtual world. Players that enter an area, query the storage infrastructure for all ephemeral data. In contrast, permanent data concerning characters should be stored locally, as it is accessed only when the player is playing in the game. Local storage ensures that is always available when needed but requires use of anti-cheating mechanisms to prevent users tampering with it. Players may backup their permanent data using the distributed storage infrastructure in case their local data becomes corrupted, but the availability for the backup is not guaranteed. Also some game world data such as the specifications and state of unique NPCs, unique magical artifacts, or enduring features of the world with variable state like Pompeii Towns before or after the eruption of Vesuvius are important to the overall story line of the game world and need to be reliably stored. In such cases both the distributed storage infrastructure and local storage can be used to ensure accessibility on demand and long term persistent storage with general availability.

3.5.3 State Persistency Discussion

Game state persistency is a major challenge for P2P MMOGs as existing P2P storage infrastructures are designed to support file sharing, and seldom fulfil the performance and security requirements of a MMOG. Compared with the previous three design issues that have been heavily researched, the persistency area is still immature with many problems waiting to be investigated.

3.6 Cheating Mitigation

In a C/S architecture, it is easier to maintain the security of an online game, because a server gives the game provider substantial control over the game, and the server is able to validate every action request sent by a client before carrying it out. However, without the existence of such an authority, prevention of cheating becomes a challenging problem in P2P MMOGs.

In the literature, many distributed cheating mitigation mechanisms have been proposed, for example [73, 107, 26, 97, 115]. Some of them attempt to prevent cheating from happening by reinforcing game event ordering and state exposure protocols, whereas some others just aim at detecting and remedying inconsistent game results after suspicious game sessions. In this section, the former category are termed Proactive Approaches, and the latter category are termed Reactive Approaches.

3.6.1 Proactive Approaches

Games involve rule-based, real-time simulations of multiple in-game objects controlled by a player [26]. A game proceeds along a sequence of time units called “frames”. In each frame, every player makes its own decision, which is an action allowable under game rules. A player can make exactly one such decision during each frame, based on

its perception and estimation about other players' status. All players' decisions are then aggregated and resolved by computing the resulting state according to game rules. The sequence of resolved states observed by each player is the game's "playout".

As discussed above, there are generally two different ways for an unscrupulous player to cheat during a game session. On the one hand, a player may gain extra advantages unfairly by peeking at current status information at other players. This form of cheating is especially common in strategy games, because it would be very helpful if a player could illegitimately discover his opponent's hidden position and activity. In this case, advanced information exposure protocols such as [41] can be applied to reinforce the fair playout of a game.

On the other hand, a more significant security flaw in a P2P MMOG is the "suppress-correct" cheat, which allows a player to gain an advantage by purposefully dropping update messages. More concretely, when a dead reckoning policy that allows n buckets to be dead-reckoned is used, a cheating player can purposefully drop $n-1$ update packets while playing. The player then uses knowledge of the current game state to construct an update packet for the nth bucket that provides some advantage [26]. For example, when player A and B are trying to shoot at each other, A can pretend to be sluggish by constantly dropping $n - 1$ updates; meanwhile B dead-reckons A's missing state but cannot confirm where A really is. For the nth bucket, A sends a fabricated update that indicates it has dodged all the bullets. However, because A receives all B's updates on time, A's shooting is not affected at all.

Lockstep [26] is the first event-ordering protocol to address fixed-delay and timestamp cheats. Lockstep orders events by rounds and increments a round only after every player has committed its move for that round. A drawback of Lockstep is that the total ordering of events suffers from the largest delay between any two players, because all other players involved in the same interaction have to wait until the commitment is received from the slowest player. NEO [73] improves this design by bounding the length of each round with a maximum latency. To prove that an event occurred at its stated time, a player

must be able to send its update to a majority of other players within the round duration. Then, voting is used to form a consensus on whether a given player has sent an update within a round. As long as a majority of players receive updates on time, normal gaming experience will not be affected by slow players. However, the tradeoff is that a player who is slow to most nearby players will not be able to play in that area of the virtual world.

Moreover, NEO requires event updates be signed and encrypted before being sent to other players, so that a player cannot modify its own action after it has learned of others' actions. In order to achieve better performance and remove potential issues with key tampering and selection, the SEA [76] protocol has been proposed to replace NEO's encryption with a cryptographic hash function as the commitment method. Recently a more efficient signature scheme EASES [42] has improved SEA by computing a message's digest before signing the message. EASES is able to prevent the existential-forage problem, as well as greatly save on computation time.

Similarly, other protocols and infrastructures have also been devised, e.g. FPS [45] and Hack-Proof [71], which actively minimize the opportunity for cheating by requiring players always to react upon the same game state information for every frame.

3.6.2 Reactive Approaches

Instead of applying sophisticated information exposure and event-ordering protocols, reactive cheating mitigation approaches just aim at detecting unfair game playouts afterwards and rolling back unfair results.

Log Auditing (LA) [97] partitions a game world into multiple regions, and a super-peer called a "region controller" (RC) is selected in each region. In LA, each player sends their commands as a signed sequence of packets, and the RC responds with signed game state updates. In this process, significant game events are logged by both the player and the RC. When given the same initial game state and player commands, the correct

output can be reproduced by rerunning the log on a trusted machine, so that cheating can be detected and unfair results can be rolled back.

Likewise, the design of [93] is based on the same idea, where the only difference is that multiple "monitor nodes" are introduced in each region. These monitor nodes calculate the latest game state from the previous game state respectively according to the game events happened during the current timeslot. Hash values of game states are compared periodically, so as to identify potential inconsistencies and cheats.

Actually, both approaches mentioned above can be classified as referee based mechanisms, in which the selection of non-colluding referees from untrusted peers is critical. Two secure referee selection algorithms, SRS-1 and SRS-2, have been proposed in [160], which not only emphasize the fairness issue, but also the communication latency among referees and common peers.

DaCAP [115] and FreeMMG [40] rely on mutual monitoring among all the players, rather than a limited number of referees. In these systems, players are organized into "legal groups" according to their locality in the virtual game world. All members of a legal group have to compute and record all actions and status of the other members in the same group. Once cheating behaviour is detected, the cheating player is reported to a "check server" with related evidence. To avoid collaboration to falsify player data, DaCAP randomly chooses a number of players from other areas of the game world to join the group, while FreeMMG allocates each group a server simulated player that can always be trusted.

Last but not least, a novel behavioural monitoring mechanism has been proposed in [107]. This approach differs from any other methods in that it does not rely on knowledge about specific vulnerabilities and their method of exploitation in order to protect the system. Instead, it relies on the real-time monitoring and analysis of players' movements and behaviours in the game world for indications of cheating play. This concept is based on the hypothesis that players engaged in cheating will exhibit behaviour which

is significantly distinguishable from normal play. If this is the case, cheaters can be identified without regard to their actual method of exploitation.

3.6.3 Cheating Mitigation Discussion

It is widely accepted that P2P MMOGs are more difficult to secure than conventional C/S architectures. Moreover, security issues present themselves at all stages in the design and implementation of P2P MMOGs. Hence it is reassuring to see that intensive research into proactive and reactive cheating mitigation mechanisms is starting to bear fruit.

3.7 Incentive Mechanisms

P2P applications are by nature voluntary resource sharing systems, in which there is often a tension between individual concerns and collective welfare. This is evidenced by well-known examples, such as Napster [28], eDonkey [155] and BitTorrent [94]. As the benefits of these systems are rooted in cooperation, they are inherently vulnerable to non-cooperative behaviour, and it is especially necessary for such systems to be designed so that participants are induced to cooperate. The mechanisms that are embedded in the system for this purpose are called Incentive Mechanisms [172].

Like other P2P applications, a P2P MMOG also requires an incentive mechanism to convince its participants to contribute their resources for the collective welfare. As discussed in the previous sections, a P2P MMOG is deployed on a collection of computing assets owned by common game participants. For example, network bandwidth is needed in game event dissemination, storage capacity is needed in game state persistency, and both CPU cycles and memory are needed in interest-management, NPC hosting and cheating mitigation. A P2P MMOG is so demanding an application that it requires considerable amount of computing resource being offered not only by super-peers, but also by every single player.

Specifically, a P2P MMOG requires two types of incentive mechanisms: accounting and reputation.

3.7.1 Accounting Mechanisms

An accounting mechanism maintains the viability of a P2P MMOG by quantifying the amount of computing resource a player has contributed to the system. On the one hand, it keeps a record of a player's historical contribution, and on the other hand, it entitles the player to consume roughly equivalent resources from other players. In this way, selfish players (a.k.a. free-riders) can be identified and discouraged, and a sufficient level of reciprocity can be ensured to make use of a P2P MMOG beneficial.

DCRC [77] is a fully distributed accounting system that can be applied to general P2P applications. The key idea in DCRC is a Debit/Credit platform using a virtual currency. By tracking a user's activities in a P2P system, DCRC bills the user according to the amount of resources that the user has consumed (i.e. Debit), and rewards the user according to the time and quality of a service that the user has offered to others (i.e. Credit). A user that stays in credit for long can be further encouraged in many different ways, e.g. service quality differentiation, application-specific wealth, or other privileges.

3.7.2 Reputation Mechanisms

Merely quantifying a peer's contribution to a P2P system is sometimes inadequate in discouraging certain disadvantageous behaviours. For example, a player may have worked as a super-peer in a region for a long time and have contributed a lot of resources to the application. However, the player may also disconnect from the system abruptly when it decides to leave, and thus put the system into an inconsistent state which takes much time and inconvenience to recover from. In this case, a reputation mechanism becomes necessary for qualifying a peer's dependability, honesty and overall manner towards P2P

collaborations, so as to make the player accountable for their positive or negative contribution to the system.

Many distributed reputation management systems can be used in a P2P MMOG. EigenTrust [98] and REPS [89] are representatives of mutual rating based approaches. In these systems, after each interaction peers produce either positive or negative feedbacks for each other, and keep the feedbacks in their own storage, i.e. the local trust values. Reputation query algorithms are provided for a peer to aggregate such local trust values from its direct friends, friends of friends, or arbitrary numbers of unacquainted peers, so that the peer can estimate approximately the trustworthiness of any other stranger peer.

In contrast, approaches like Proactive Reputation [152] and Local Reputation [116] do not depend on ratings from third parties. Instead, they provide various means for a peer to evaluate the trustworthiness of a target peer directly, hence they are inherently immune to bad-mouthing or collusion attacks. These approaches focus on addressing the challenge of anonymous reputation requests inside application traffic, because once an unscrupulous peer determines that the purpose of a request is to measure its reliability, it will always process these requests correctly to boost its reputation.

Finally, some P2P applications are vulnerable to the so called "white-wash" problem, i.e. selfish and irresponsible past behaviour is hidden by acquiring a new online identity that is not burdened with a record of past misdemeanours. However, in P2P MMOGs, the problem is naturally avoided, because the MMOG features expensively acquired identities. Though an unscrupulous player is able to switch to a new Id at any time, the player will lose all in-game characteristics like experience levels that are bound to the previous user account and cannot be transferred to the new one. So, "new participants" can be more safely granted an initial reputation and contribution credit in order to help them start playing smoothly.

P2P MMOG Architectures	Interest Management	Event Dissemination	Task Sharing	Game State Persistency	Incentive	Overall Evaluation
P2P Support '04	Region-based	ALM	Static	None	None	Basic
Distributed '04	Region-based	Unicast	Dynamic	Distributed	None	Refined
OPeN '05	Spatial	Unicast	None	Centralized	None	Refined
P2P Arch '06	Region-based	ALM	None	PAST	None	Basic
VAST '07	Voronoi	Unicast	Dynamic	None	REPS	Advanced
Mediator '07	Hybrid	Unicast	Dynamic	Distributed	DCRC	Advanced

Table 3.1: Comparison of representative P2P MMOG architectures

3.7.3 Incentive Discussion

The success of a P2P MMOG relies on an effective incentive mechanism that facilitates the collection of resources, and a reputation mechanism that minimises antisocial behaviours. Such incentive mechanism is a key design issue that is often unjustly ignored in the literature. As we shall see in the next section some P2P MMOG infrastructures may need to improve their incentive mechanisms to make them more practical.

3.8 Comparison of P2P MMOG architectures

In this section, five pieces of related work [118, 59, 57, 79, 87], as well as the Mediator framework [64] proposed in this thesis, are selected as representatives of recent P2P MMOG architectures. Table 3.1 summarizes their features and illustrates how they address the essential issues discussed previously (related work is ordered according to its time of publication). However, cheating mitigation is ignored in this table, because it is a relatively separate issue, and all the architectures are potentially compatible with existing cheating mitigation techniques.

1) *P2P Support '04* [118] This early architecture partitions a virtual game world into multiple regions, and interest-management is carried out by region-based publish/subscribe. Each region is associated with a publish channel, to which all region participants subscribe, and gaming events are delivered using Scribe. A super-peer called a "region coordinator" is selected in each region, which hosts all the NPC objects in that region. A prototype application, "SimMud", is implemented.

While representing a number of good design decisions, this architecture is evaluated as *basic* for the following reasons. The IM scheme is coarse-grained and its event dissemination relies on general purpose ALM middleware, which may induce high communication latency. Also, its NPC host allocation mechanism is intuitive, and neither specific game state persistency nor incentive mechanisms are provided.

2) *Distributed '04* [57] This architecture adopts a region-based interest management scheme, where the size of each region is quite small. In this case, the number of players in a region is limited, so players in the same region can communicate with each other using unicast. However, it is likely that a gaming event that takes place in one region may also affect the players in neighbouring regions. Therefore, a super-peer is selected in each region to propagate local gaming events to neighbouring super-peers when necessary. Furthermore, the architecture suggests that each player stores its own permanent data, while public ephemeral data is stored by a Distributed Hash Table [150].

The architecture is evaluated as *refined*, because its IM, event dissemination and state persistency mechanisms are well designed. However, it is only held to be *refined* because it assumes that there are always adequate peers donating computing resources, and thus when a NPC becomes active, a random capable peer is selected to host that NPC. Such NPC host allocation and incentive mechanisms are overly simple.

3) *OPeN '05* [59] The OPeN architecture proposes a distributed spatial data index service, which is built on top of a structured P2P overlay network. With this service,

players can register their current locations in a game world, and query about other entities in their AOIs. Nearby entities establish direct UDP packet flows with each other in order to exchange gaming events. Persistent game data is stored and managed by a centralized database server. A simple P2P MMOG is implemented for demonstration purposes.

The architecture is evaluated as *refined*, because its IM and event dissemination mechanisms seem to be adequate. However, it is only held to be *refined* because the infrastructure still depends on game servers for state persistency, and NPC host allocation and incentive mechanisms are not supported.

4) *P2P Arch '06* [79] This architecture is purely Pastry based, as it uses Scribe for game event dissemination, and PAST for game state persistency. The architecture employs coarse-grained region-based interest-management, but does not provide details about NPC host allocation and incentive mechanisms.

The architecture is evaluated as *basic*, because it directly employs ALM and distributed storage middleware built on top of Pastry. As these middleware components are designed for general P2P applications their performances may not be adequate for a P2P MMOG. No arguments are provided for the suitability of the middleware nor demonstrations of its effectiveness.

5) *VAST '07* [87] A unique Voronoi assisted interest-management mechanism is employed in the VAST project. The Voronoi diagram is also inherently suitable for virtual distance based NPC host allocation. Furthermore, the architecture provides a native incentive mechanism, namely REPS [89]. A prototype application "ASCEND" is implemented.

Though currently the architecture still requires a game server for peer bootstrapping, load-balancing, fault-tolerance and state persistency purposes, it is evaluated as *advanced*, because its application of the Voronoi technology is quite remarkable, which

offers a consistent way of fine-grained IM, efficient game event dissemination and convenient NPC host allocation.

6) *Mediator '07* [64] The Mediator framework employs a hybrid IM scheme like MOPAR [168] and disseminates gaming events through unicast communication. A novel NPC host allocation mechanism, Deadline-Driven Auctions (DDA), is devised to support the sharing of real-time NPC tasks. DDA is inherently compatible with reactive cheating mitigation approaches, e.g. Log Auditing [97], and also supports a DCRC-like [77] incentive mechanism that motivates application participants to contribute their resources to the system. Furthermore, a membership-aware multicast mechanism (MAMBO) [65] is developed for maintaining game zone infrastructures efficiently. It is also convenient to support game state persistency using PAST [144], as both PAST and MAMBO use the same overlay network. Key components of this framework as well as a test-bed application are implemented.

The architecture is evaluated as *advanced*, because it addresses all of the design issues surveyed by this chapter. In particular, its NPC host allocation mechanism is good at minimising communication latency among NPC hosts and ordinary players. More details about the Mediator framework will be given in Chapter 4.

3.9 Conclusion

Over recent years, substantial research efforts have been devoted to adapting MMOGs to P2P architectures. A crucial step of this PhD research is to develop an in-depth understanding of the related work by clarifying their objectives, discovering their relationships, comparing their differences, and most importantly analysing their advantages and limitations. In this process, six key design issues for P2P MMOGs have been identified, including interest management (Section 3.2), event dissemination (Section 3.3), task sharing (Section 3.4), state persistency (Section 3.5), cheating mitigation (Section 3.6),

and incentive mechanisms (Section 3.7). This chapter systematically discusses design alternatives for each issue, and evaluates to what extent these issues have been addressed by existing P2P MMOG architectures.

Among the six key issues, interest management and event dissemination mechanisms have been heavily studied, with abundant practical technologies proposed in the literature. By comparison, state persistency and cheating mitigation are developing research areas, which have already started to bear fruit but still have some problems waiting to be solved. However, it is important to notice that few existing architectures have addressed task sharing and incentive mechanisms appropriately, although they are also essential aspects to the success of a P2P MMOG. This situation will be improved with a new comprehensive design framework that is presented in the next chapter.

— Nothing is particularly hard if you divide it into small jobs.

Henry Ford

Chapter 4

Mediator Framework

4.1 Introduction

This chapter presents the design of Mediator, a framework for peer-to-peer massively multi-player online games. It is a framework because it provides a generic architecture that is flexible and extensible for application developers to customise according to their actual needs. The aim of this framework is to address the key design issues identified in Chapter 3 within an integrated system, so as to offer key elements of a comprehensive and feasible blueprint for P2P MMOG applications.

Section 4.2 introduces the Mediator framework from various perspectives at a high level. Section 4.3 further elaborates on the responsibilities and self-organising selections of four major super-peer roles. However, a super-peer runs on an application participant's machine, whose availability and dependability are not comparable to a dedicated game server running on a dedicated platform. So, Section 4.4 discusses exceptional states that may be caused by super-peer failures, and corresponding mechanisms for a P2P MMOG to recover from these accidents automatically. In particular, a hierarchical supervision architecture is presented at the end of the section, which could be facilitated by the

MAMBO technology that is presented in Chapter 5. Finally, Section 4.5 provides a critical analysis of the strengths and weaknesses of the Mediator framework in comparison with conventional and P2P MMOG architectures.

4.2 Mediator Overview

4.2.1 The Three Conceptual Layers

As MMOGs have scaled up rapidly, conventional C/S architectures have started to exhibit various technical and commercial drawbacks, primarily reliability and scalability costs, as discussed in Section 2.3.1. Hence several P2P MMOG architectures have been proposed in the literature [139, 108, 91, 87, 57, 79, 118], aimed at reducing the costs of providing and maintaining a game server infrastructure by exploiting game participants' computing resources while maintaining a sufficient level of reliability. In comparison with the related systems, ***Mediator*** is an integrated design framework that addresses all of the six key technical challenges of P2P MMOGs. The framework comprises three conceptual layers, as depicted by Figure 4.1:

1. Structured P2P Overlay The characteristics and functionalities of a structured P2P overlay, e.g. CAN [134], Chord [150] and Pastry [143], have been discussed in Section 2.4.3. The Mediator framework uses a P2P overlay as the rendezvous point for a MMOG application. Concretely, in order to join a MMOG, an intending participant needs to know the IP address of at least one existing peer in the system (referred to as the "linkman"), which can be either a well-known public peer or a friend who is already playing the game. The intending participant will identify itself with a random PeerId and exchange several messages with its linkman according to the overlay's working protocol to join the overlay. Once a participant has joined, it will be able to deliver messages to any destination Ids using a routing algorithm that is provided by the underlying overlay

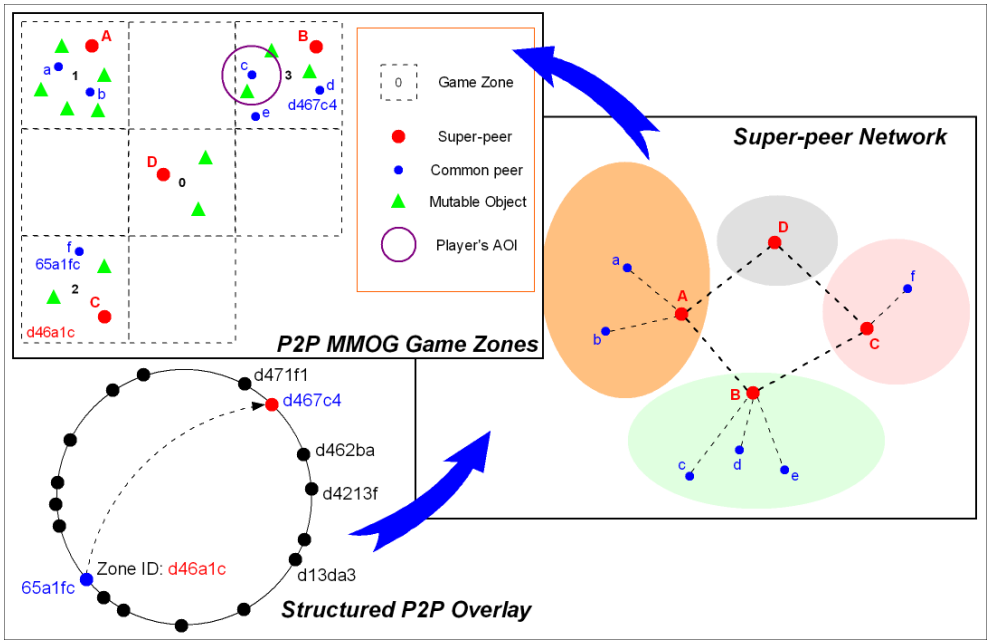

Figure 4.1: Three conceptual layers in the Mediator framework

infrastructure. The bottom left part of Figure 4.1 illustrates a situation where a number of application participants have joined a Pastry overlay and been organised into a logical ring.

2. Super-peer Network In the P2P overlay, certain application participants are selected to carry out various computational and administrative tasks according to predefined selection criteria. These peers are called "super-peers", and are entitled to govern other ordinary peers. Generally speaking, a super-peer is a node in a P2P network that acts both as a service provider to a set of clients, and as an equal in a network of super-peers [166]. The super-peer selection problem has been identified across a spectrum of P2P applications. Basically, the problem involves the selection of a subset of the peers in a large scale, dynamic network to serve in distinguished roles, and sometimes those selections also have to satisfy additional requirements, such as suitable allowance for load-balancing and fault-tolerance [117, 124]. The right part of Figure 4.1 illustrates a scenario where some super-peers have been selected, controlling a number of ordinary application participants. Here, a simplified notation is used for demonstration purposes.

Actually, the Mediator framework features four kinds of primary super-peer roles, whose responsibilities and interactions are discussed in the next section.

3. P2P MMOG Game Zones The Mediator framework adopts a divide-and-conquer strategy and partitions the whole virtual game world into multiple consecutive quadrant zones. A game zone is the largest granularity of game space for super-peer selection and game logic execution. According to the current design, each game zone comprises one boot mediator, one zone mediator, one interest-management mediator, an adequate number of resource mediators, and multiple ordinary player peers. For load-balancing purposes, it is possible for a crowded zone to be further partitioned into multiple sub-zones, and similarly, multiple contiguous vacant zones to be merged into one larger vacant zone. Every game zone is identified by a ZoneId. The main difference between a PeerId and a ZoneId is that the former is random and temporary, whereas the latter is static and well-known by all participants of the application. In the P2P overlay, a ZoneId serves as a rendezvous point for the peers that are playing in that game zone. The upper left part of Figure 4.1 illustrates a virtual game world that is composed of eight game zones.

4.2.2 The Four Super-Peer Roles & Their Interactions

Currently, the Mediator framework comprises four super-peer roles: Boot Mediator (BM), Zone Mediator (ZM), Interest-Management Mediator (IMM) and Resource Mediator (RM). Figure 4.2 outlines the following ways in which the super-peers may interact with each other, as well as with ordinary players:

Interactions among super-peers & ordinary players This category of interactions is required by the normal play of a P2P MMOG and is represented by solid arrows in Figure 4.2. The circular arrow in the center of the figure further indicates the order in which an ordinary player initiates such interactions during a typical game session. First

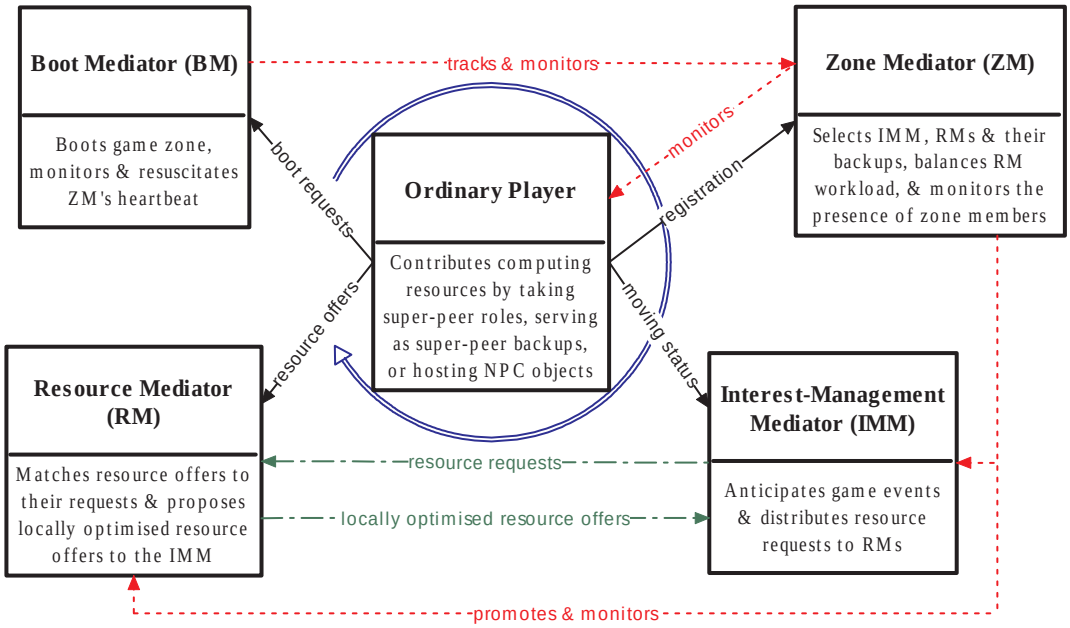

Figure 4.2: Super-peer roles & interactions

of all, to join a P2P MMOG a player needs to route a boot request to the BM for a target game zone. A BM is a super-peer that provides a look up service in a P2P overlay network and replies to boot requests with the communication endpoint of the current ZM for that zone. After the bootstrapping process, the player registers at the corresponding ZM to become one of its zone members. A ZM is responsible for establishing the super-peer infrastructure in a game zone with resource rich peers. Also, a ZM notifies its zone members about the super-peers that it has selected (i.e. the IMM and RMs) via periodical heartbeat messages [141]. An IMM is by nature a game event anticipator. Ordinary players are required to inform the IMM about their moving states. A RM, on the other hand, serves as a resource matchmaker that proposes appropriate resource providers to the IMM for hosting NPC objects according to game scenarios. Multiple RMs may coexist and work in parallel in a game zone, and ordinary players are able to upload their resource availability information to the RMs periodically. More details about super-peers' responsibilities and selections will be discussed in Section 4.3.

Interactions for reinforcing super-peer dependability Because the robustness and availability of a super-peer in a P2P MMOG are not comparable with a dedicated game server, fault-tolerant mechanisms are needed to enable a P2P MMOG to recover from various exceptions automatically. The interactions that are carried out for this purpose are represented by dashed arrows in Figure 4.2. In short, a hierarchical supervision relationship is established, in which the BM monitors the working status of the ZM, and the ZM in turn monitors the IMM, RMs and ordinary zone members. Further discussions about super-peer dependability and the supervision architecture will be given in Section 4.4. Also, an efficient supervision technology, Membership-Aware Multicast with Bushiness Optimisation (MAMBO) that is primarily devised for the Mediator framework will be presented in Chapter 5.

Interactions among the IMM and multiple RMs This special category of interactions is represented by dashdotted arrows in Figure 4.2. The purpose of these cooperations is to address the NPC host allocation problem discussed in Section 3.4. As a game event anticipator, the IMM has a global view of the game zone and is able to forecast the creation or activation of NPC objects. Accordingly, the IMM distributes resource requests to the RMs in advance, asking for appropriate NPC hosts. On the other hand, the RMs aggregate resource availability information from ordinary players, match existing resource offers to their requests, and reply to the IMM with locally optimised resource offers. This process is referred to as Deadline-Driven Auctions (DDA), and will be discussed in detail in Chapter 6.

4.2.3 Addressing the Key Design Issues

The Mediator framework accounts for the key design issues for P2P MMOGs in the following ways:

Interest-Management The Mediator framework adopts a MOPAR-like hybrid interest management scheme that is discussed in Section 3.2.3. In every game zone, a dedicated super-peer (a.k.a. Interest-Management Mediator, or IMM) is selected to work like a Master Node in MOPAR [168]. Ordinary player peers in the same game zone are required to update the IMM when their moving states are changed, so that the IMM can obtain a global view of its game zone and anticipate the location of the players in the near future. The IMM is able to notify related players about forthcoming gaming events, as well as to organise the spawning of NPC objects as required by game scenarios.

Event Dissemination Unicast communication is used for exchanging gaming events among player peers that are involved in an interaction. Because the hybrid IM scheme endows a player with information about other players or NPCs that are about to show up in its AOI, the player only needs to establish direct P2P connections with a limited number of other players or NPC hosts when it becomes necessary.

Task Sharing Mediator's distributed hosting of NPC objects is based on a distinctive task mapping mechanism, Deadline-Driven Auctions (DDA). DDA is primarily designed to support the sharing of real-time NPC tasks in a large-scale P2P MMOG, though it may also apply to general P2P applications that feature computational and interactive tasks with various deadlines. The system model of DDA involves three different parties, which are a set of resource providers, a work source, and an adequate number of matchmakers. In the Mediator framework, an IMM serves as the work source, as it constantly initiates the generation of NPC tasks according to the game scenario. In addition, ordinary player peers in a game zone are resource providers, which have spare computing resource available on their machines. In order to bridge between such resource availability and resource requirement, dedicated super-peers (a.k.a. Resource Mediators, or RMs) are selected to supply the role of matchmakers. More details about the design, analysis, implementation and evaluation of DDA are provided in Chapter 6.

State Persistency It is fairly convenient to support game state persistency in the Mediator framework using the PAST [144] distributed storage middleware that is discussed in Section 3.5. As we shall see in Chapter 5, Mediator relies on Membership-Aware Multicast with Bushiness Optimisation (MAMBO) for efficient game zone structure maintenance. MAMBO is in nature an enhanced version of the Scribe application-level multicast service [38], and both Scribe and PAST are built upon the Pastry P2P overlay infrastructure [143]. Hence, it is reasonable to apply Scribe and PAST simultaneously in the same P2P MMOG.

Cheating Mitigation The NPC task sharing mechanism used by the Mediator framework is inherently compatible with reactive cheating mitigation approaches, e.g. Log Auditing [97] that is discussed in Section 3.6.2. Unlike virtual distance based mechanisms, DDA allocates NPC tasks to third party hosts which are able to provide adequate computing resources, as well as to guarantee acceptable game interactivity. So, it offers the opportunity for a reactive cheating mitigation scheme to be applied to reinforce the fairness of a PC-vs.-NPC interaction. PC-vs.-PC interactions can also be secured by proactive cheating mitigation approaches like NEO [73] and EASES [42] that are discussed in Section 3.6.1.

Incentive Mechanism The Mediator framework incorporates native support for DCRC-like [77] incentive mechanisms. DDA incorporates a reward scheme that keeps a record of the resources that a peer has contributed to the system, and accordingly, to entitle the peer to consume roughly equivalent resources from other peers. In this way, selfish peers can be identified and discouraged. Furthermore, DDA also supports flexible matchmaking policies, and with a friendly incentive policy, can establish a cooperative economic model that shares NPC tasks more fairly and helps motivate application participants to contribute their resources to the system.

4.2.4 Key Characteristics

The Mediator framework is designed to exhibit the following characteristics:

Self-organisation The Mediator framework is self-organising, as its infrastructure can be assembled and maintained automatically. With the help of a structured P2P overlay, an application participant only needs to know minimal information (i.e. the communication endpoint of an existing peer) and to carry out minimal configuration (i.e. identifying itself with a random PeerId) in order to join the system. As the system evolves, super-peer roles (Section 4.3) and NPC tasks (Chapter 6) can also be distributed to suitable participants automatically. A P2P MMOG does not have any centralised game servers and its users log in and out of the game unpredictably, so self-organisation is an important characteristic that is needed to adapt the system to fluctuating resource availability.

Scalability A crucial reason for deploying a MMOG using a P2P architecture is to pursue better scalability, and it is also a principle underlying the design of the Mediator framework. In order to bring the potential of a P2P system into full play, the Mediator framework puts much emphasis upon the utilisation of incentive mechanisms to convince application participants to contribute their computing resources to the system and to facilitate a sufficient level of reciprocity. As a result, free-riders can be identified and discouraged (Section 6.3.3). More users will always bring more resources to a P2P MMOG and to the right places, because more players coming to a zone will bring more resources to that zone. On the other hand, the framework is designed to make effective use of available resources. For example, it distributes a large amount of NPC tasks to ordinary participant machines. Furthermore, with suitable super-peer load-balancing mechanisms and the use of sub-zoning techniques, the Mediator framework has a good potential to support large scale MMOG applications with hundreds of players per zone (Section 6.6) over indefinitely many zones.

Dependability In a P2P MMOG, a game participant is mainly responsible for its own playing of the game, plus the hosting of a limited number of public tasks. Therefore, the failure of an individual participant will only affect a few other players in a short period of time, instead of cutting out the normal play of all the other players in the game. However, a super-peer in a P2P system can be expected to be less dependable than a dedicated game server in terms of network connectivity, software and hardware capacity. Therefore, the Mediator framework is designed for resilience and is able to recover from various exceptional states. First of all, a super-peer's dependability is enhanced by a number of super-peer backups and data replication, as described in Section 4.4. The failure of a working super-peer can be detected in due course, and one of the backups will step up to replace its predecessor automatically. This backup mechanism enables Mediator's super-peer infrastructure to be as dependable as the underlying P2P overlay. In addition, a MAMBO technique (Chapter 5) has been devised to establish a hierarchical supervision architecture within a game zone. MAMBO reuses the communication structure of a tree-based ALM system and helps to track the presence of application participants efficiently.

Integrity Table 3.1 in Section 3.8 compares the characteristics of six existing P2P MMOG architectures, among which the Mediator framework is evaluated as *advanced*, because the design takes into consideration each of the six challenges in the design of a P2P MMOG. In comparison, most of the other related work only addresses some of the key design issues identified in Chapter 3, and thus does not identify a sufficiently comprehensive way for adapting a conventional MMOG to a P2P architecture. This thesis endeavors at presenting a more complete and feasible framework that could serve as a blueprint for designing practical P2P MMOG applications.

Flexibility & Extensibility It is important to notice that the design of the Mediator framework is flexible and extensible. On the one hand, it is flexible because even though useful technologies such as MAMBO and DDA have been proposed to fulfil specific

requirements of the framework, they are not tightly coupled with the framework. For example, DDA is applicable to general P2P applications that feature computational and interactive tasks with various deadlines, and it is also possible for the Mediator framework to allocate NPC hosts using a different approach. Similarly, it has been mentioned that MOPAR [168] is adopted as the IM scheme for Mediator, but it does not mean that Mediator is only compatible with adopting MOPAR. On the other hand, the framework is extensible as new super-peer roles can be easily introduced into the framework according to newly identified requirements. For example, currently security is not a primary focus of the Mediator framework. However, it can be imagined that a set of trusted super-peers could be present in the system for authentication, authorisation and accounting (A^3) purposes [93, 40]. The framework is open for modification and extension, and may collaborate with peripheral mechanisms that are implemented to facilitate various P2P functionalities.

General characteristics of P2P applications The Mediator framework is designed to exploit general characteristics of P2P applications of being low cost to maintain and run, of being suitable for collective implementation on the open source model, and through engaging the interest of a community of player-developers, of being more likely to last than rival commercial services.

4.3 Mediator Super-Peer Roles and Selection

This section elaborates on the responsibility and self-organising selection of each class of super-peer in the Mediator framework, including the Boot Mediator for handling bootstrapping messages (Section 4.3.1), the Zone Mediator for maintaining the super-peer infrastructure (Section 4.3.2), the Interest Management Mediator for anticipating game events (Section 4.3.3), and the Resource Mediator for locating appropriate resource providers to host NPC objects (Section 4.3.4).

4.3.1 Boot Mediator

A Boot Mediator (BM) is the rendezvous point of a game zone, and is responsible for handling bootstrapping messages sent by other peers that are about to join that zone. The selection of a BM is based on Id space locality, rather than the game world locality in terms of geography. Specifically, a BM is a super-peer whose PeerId is numerically closest to a ZoneId in the P2P overlay. As a result, a BM is not necessarily a member of the game zone that it works for. This is an important difference between the BM and other super-peer roles.

A player who wants to join a P2P MMOG needs to carry out the following procedures:

- Firstly, the player creates a random PeerId (e.g. a Pastry Id) to identify itself during this game session.
- Then the player sends out a boot request with a ZoneId that specifies the game zone it wants to join.
- The boot request is delivered by the overlay network to an existing peer in the system, whose PeerId is numerically closest to the ZoneId.
- The peer that receives the boot request is selected to be the BM for that zone, which processes the request and responds to the sender with a boot reply.

Here, a concrete example is helpful to clarify a BM's selection process and responsibility. Suppose that in the bottom left part of Figure 4.1, peer $65a1fc$ is attempting to join zone $d46a1c$. In this case, the peer sends out a boot request towards the destination Id $d46a1c$. This message is finally delivered to peer $d467c4$ by the overlay network, because at present its PeerId is numerically closest to the destination Id. As a result, peer $d467c4$ is selected to be the BM for zone $d46a1c$. If peer $65a1fc$ is the first one to join that zone, BM $d467c4$ responds with a promotion message that allows the peer to become a provisional Zone Mediator (ZM). Otherwise, the BM replies to the peer with

the communication endpoint of the existing ZM for that zone. As we shall see in the next section, a ZM is responsible for zone structure maintenance, and every new zone member is required to register its presence at the ZM. However, as peers join and leave the system, the ZM for each zone may change over time. Because a P2P MMOG lacks centralised control, it is difficult for newly joined peers to get in touch with the latest working ZM. From this point of view, a BM's main responsibility is to provide a look up service that helps newly joined peers to find their ZMs.

The previous example also demonstrates that the selection of a BM is fully self-organising, as an application participant only needs to identify itself using a random PeerId before joining a P2P MMOG, and only needs to know minimal information about the game world, namely an Id for the zone he wishes to enter. Then, a BM can be determined automatically according to the routing algorithm of the underlying P2P overlay. Furthermore, because a BM promotes the first member of a game zone to be a provisional ZM, which in turn establishes the rest of the super-peer infrastructure, the self-organisation property of the Mediator framework is rooted in self-organising selection of BMs.

However, this super-peer selection strategy is subject to the randomness of the topology of a P2P overlay network. In other words, because every application participant identifies itself using a random PeerId, it is out of the application developers' control which participant will be selected as a BM. Consequently, two significant issues have presented themselves in the design of the Mediator framework:

1. Separation of super-peer responsibilities Due to the randomness in the selection process, a BM is selected regardless of a peer's actual resource availability. In this case, an important rationale behind the design of the Mediator framework is to separate super-peer responsibilities appropriately. In particular, the services to be performed by a BM must not be arduous or time sensitive, so that any ordinary peer qualifies for this role. Although some related systems [118, 91, 168, 79] also feature self-organising super-peer selection mechanisms, they suffer from the crucial defect of imposing too much workload on a randomly selected super-peer, e.g. hosting NPCs for the entire zone. In

contrast, the Mediator framework either distributes the workload to large numbers of application participants (Chapter 5 & 6), or only assigns demanding tasks to resource-rich super-peers (Section 4.3.2). From this perspective, the Mediator framework is more efficient in terms of resource utilisation, and its design is more comprehensive and feasible than the related work.

2. Even distribution of ZoneIds Another practical issue related to the random selection of BMs is the even distribution of ZoneIds. Suppose that a P2P MMOG comprises 10 game zones, and an extreme example for unevenly distributed ZoneIds could be 10 continuous Ids from $d46a10$ to $d46a1a$. Consequently, a peer whose PeerId is $d46a15$ may be selected as the BM for all the 10 game zones. To avoid this from happening the designer of a P2P MMOG should guarantee that all the ZoneIds are carefully arranged across the whole Id space. In this case, there is only a small probability for randomly generated PeerIds to be unevenly distributed, which may cause the same peer to be selected as BMs for multiple zones. However, the precondition for such an unusual problem to happen is an adequately small online population, so even though a peer is required to handle boot requests for multiple zones, the peer is unlikely to be overloaded.

Figure 4.3 is a UML diagram that specifies a BM's activities, including the processing of boot requests as discussed above. Besides this, there are also other activities that are not covered by this section, such as K backup, periodical probing of potential new BMs, and the monitoring of a ZM's heartbeat. These activities are carried out by a working BM and its neighbouring backup peers to enhance the dependability of a P2P MMOG, and will be discussed in Section 4.4.1.

4.3.2 Zone Mediator

A ZM is an important super-peer role that is responsible for establishing the super-peer infrastructure within each game zone, as well as roughly balancing out the workload among multiple super-peer instances. Furthermore, a ZM also monitors the working

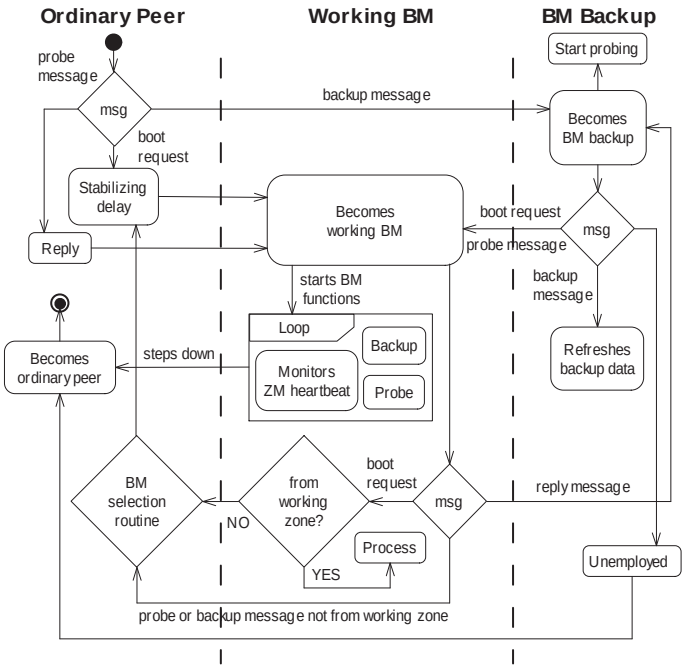

Figure 4.3: Boot Mediator activity diagram

state of the Interest-Management Mediator (IMM) and Resource Mediators (RMs) in its game zone, and maintains the robustness of the system by replacing failed super-peers with capable backups in good time.

Compared to a BM that is used to process trivial enquiry messages, the workload of a ZM may be heavier, and thus a ZM is preferably a peer with abundant processing power and network bandwidth. So, the selection of a ZM comprises the following stages:

- The first member of a game zone is promoted to be a ZM in spite of its actual resource availability, because at that time the zone is empty and there is not much work to do.
- As more peers join the zone, the first ZM's workload increases gradually, until it perceives the risk of being overloaded. During this time, other zone members will have completed their local scheduling activities, and resource-rich peers may register at the first ZM as super-peer backups. Local scheduling is a part of Deadline-Driven Auctions, which is discussed in detail in Section 6.3.3. Once a suitably

capable super-peer backup is identified, the first ZM will give up its position to that backup, which in turn becomes a formal ZM.

- Such a handover process may occur repeatedly, either as the population of a game zone keeps scaling up, or when a ZM has to leave the system.
- Where an overcrowded zone induces too much workload to be handled by any individual peer, the working ZM may divide the zone into four sub-zone quarters and appoint ZMs for each of them from super-peer backups. When the four quadrants of a partitioned zone become sparsely populated again, they will run an election protocol to replace themselves with one of their number for all four quadrants.
- When the last player leaves a zone, the player will wait a suitable period before informing the BM that the ZM for that zone has become redundant and is terminating.

A ZM has four important administrative duties, which are described below:

1. Super-peer promotion At present, the Mediator framework comprises four super-peer roles, which are BM, ZM, IMM and RM. As discussed previously, the selection of a BM is self-organising, and a BM promotes the first member of a game zone to be the first ZM, which in turn gives up its position to a formal ZM. Finally, a ZM is in charge of selecting the IMM and multiple RMs to complete the super-peer infrastructure. Both the IMM and RMs are promoted from resource-rich volunteer super-peer backups that have registered at the ZM.

2. Super-peer monitoring A ZM monitors the working state of the IMM and RMs that it has promoted, and once any of these super-peers have failed due to network connection, software or hardware reasons, the ZM will replace them with other super-peer backups immediately. This monitoring functionality is carried out using MAMBO technology that is discussed in Chapter 5. A ZM sends heartbeat messages to all its zone

members periodically via application-level multicast, which will be explained soon. MAMBO is able to reuse this multicast infrastructure to establish hierarchical supervision relationships among information forwarders and receivers. So, even though a ZM is required to track the presence of all the zone members, it can be done with little effort.

3. Super-peer load-balancing Because the current design of the Mediator framework uses a single ZM and IMM in each game zone, their load-balancing is mainly achieved through dynamic zoning. In other words, when a zone is overcrowded, its ZM may divide the zone into several sub-zones and appoint new ZMs and IMMs for each of them. However, because the DDA mechanism discussed in Chapter 6 supports multiple RMs to work in parallel, a ZM can balance out their workload by promoting an adequate number of RMs and assigning roughly equal number of ordinary peers to each of them. More details are discussed in Section 4.3.4.

4. Heartbeat A ZM periodically informs its zone members about the current status of the game zone, including a manifest of all existing zone members, communication endpoints of the IMM and available RMs, and the requirement for super-peer backups. This is implemented as regular heartbeat messages. Firstly, a manifest is provided for ordinary peers to carry out their local scheduling activities. Secondly, communication endpoints are provided for newly joined peers to get in touch with related super-peers, and also for existing zone members in case some super-peers have been replaced. Finally, the super-peer requirement is provided for ordinary peers to evaluate their own resource availability, so that qualified peers may volunteer as super-peer backups. Compared to real-time game events, such zone status information is less emergent and can be disseminated via application-level multicast, e.g. Scribe that is discussed in Section 3.3.2, so as to reduce communication overheads.

Figure 4.4 shows a detailed registration process between an ordinary peer and a ZM. Through the bootstrapping process an ordinary peer finds out the communication endpoint of the latest ZM from the BM for that zone, and then the peer registers its presence

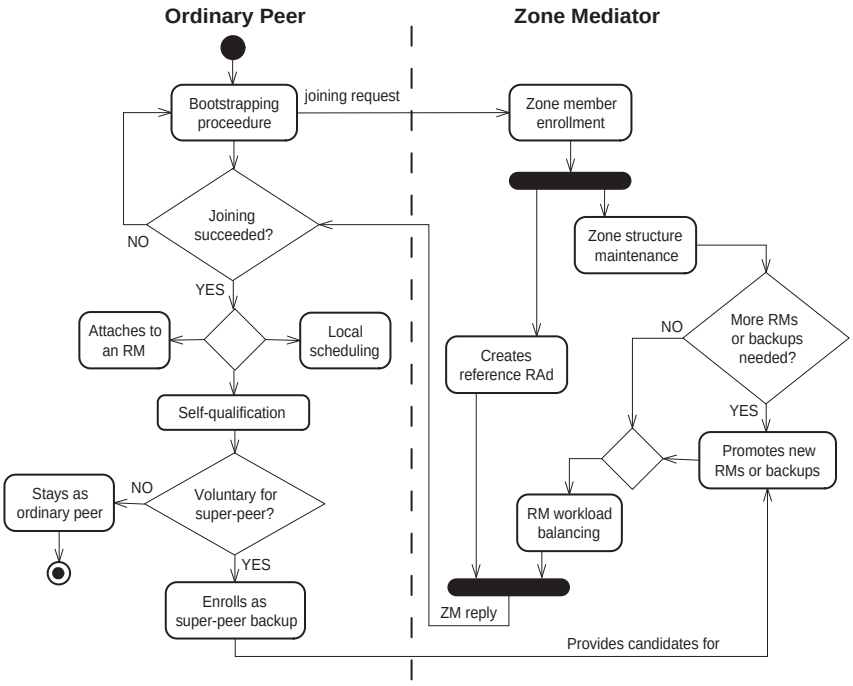

Figure 4.4: Zone Mediator activity diagram

with the ZM using a joining request. The acceptance of a new zone member may cause the zone population to reach its upper limit, so the ZM firstly carries out a series of zone structure maintenance procedures, analysing the workload of each super-peer role, deciding whether it is necessary to divide the zone into sub-zones, and whether more super-peers and backup peers are required. To ensure the performance of a NPC host allocation mechanism (Chapter 6), a ZM also attempts to keep the number of ordinary peers and RMs to a relatively stable ratio, and to balance the workload for each RM by only telling the addresses of lightly-loaded RMs to new zone members. After these procedures, the ZM replies to the new zone member with a message that contains the addresses of the current IMM and available RMs, together with the super-peer selection criteria, i.e. the "reference RAd" (Section 6.3.3). Accordingly, the ordinary peer starts to inform the IMM about its moving status, to upload its resource availability information to one of the RMs, to evaluate its own hardware configuration and to decide whether to volunteer for super-peer jobs or super-peer backups.

4.3.3 Interest-Management Mediator

As indicated by its name, an IMM is responsible for the interest-management job and serves as a game event anticipator. Because the Mediator framework adopts a MOPAR-like hybrid IM scheme, the role of an IMM is similar, but not limited to a Master Node in MOPAR.

In each game zone, a single IMM is selected by the ZM from super-peer backups, as described in the previous section. Once an IMM is chosen, its communication endpoint in the P2P overlay network will be disseminated to all the zone members in the next heartbeat message from the ZM. Accordingly, zone members start updating their positions and moving states at the IMM. In order to minimise the amount of such update messages, a peer may only notify the IMM when its moving direction and velocity is changed, and the IMM will estimate the peer's course by itself. Synchronization of a peer's precise position in a game world can be carried out when it is necessary. In this way, the IMM is able to obtain a global view of its game zone and to anticipate forthcoming game events.

It is important to clarify that the hybrid IM algorithm mentioned above is not an original contribution of this PhD research, and thus its details are beyond the scope of this thesis. However, as such algorithms have been demonstrated to be effective on anticipating PC-vs.-PC game events [168, 140], we assume that the algorithms will be equally effective in anticipating PC-vs.-NPC game events. In other words, if an IMM is able to forecast that a player character is about to enter another player character's AOI, it should also be able to forecast that a non-player-character is about to enter a player character's AOI in a similar way. Hence, the IMM is regarded as a major source of NPC tasks in the design of Deadline-Driven Auctions in Section 6.3.4.

4.3.4 Resource Mediator

A RM is essentially a resource matchmaker that facilitates the DDA NPC host allocation mechanism, which is the theme of Chapter 6. So, more details about the cooperation among the IMM and RMs will be discussed later. Here, it is important to notice that unlike the ZM and IMM, multiple RM instances may coexist and work in parallel in each game zone.

A RM is responsible for maintaining two queues: a resource queue and a task queue. The resource queue is used by a RM to cache resource availability information received from ordinary game participants. On the other hand, the task queue is used to cache NPC tasks the RM receives from the IMM. A RM's resource queue cannot be arbitrarily long due to performance considerations. In other words, a RM cannot accept and handle resource availability information from too many application participants. So, the ZM for a game zone should promote adequate RMs from super-peer backups and make sure that each of them takes charge of roughly equal numbers of zone members.

The first requirement can be satisfied by keeping the number of RMs and the zone's population to a stable ratio. Every time a new player joins the zone, the ZM analyses whether a new RM is needed. The second requirement is also known as the RM load-balancing issue, which can partly be solved by the ZM's heartbeat mechanism. A ZM exposes the communication endpoints of current IMM and RMs to its zone members through heartbeat messages, as explained in Section 4.3.2. A newly joined zone member can choose one available RM from the heartbeat, to which it uploads resource availability information. When a RM's resource queue is full, the RM notifies the ZM to remove its communication endpoint from later heartbeats, so it can be temporally hidden from newly joined players. However, it is still possible that the RM has been chosen by some players according to the last heartbeat. In this case, the RM explicitly rejects their resource information and tells them to try with other RMs. As players leave the game zone, some previously busy RMs may become lightly loaded again and restore their communication endpoints in ZM's heartbeats.

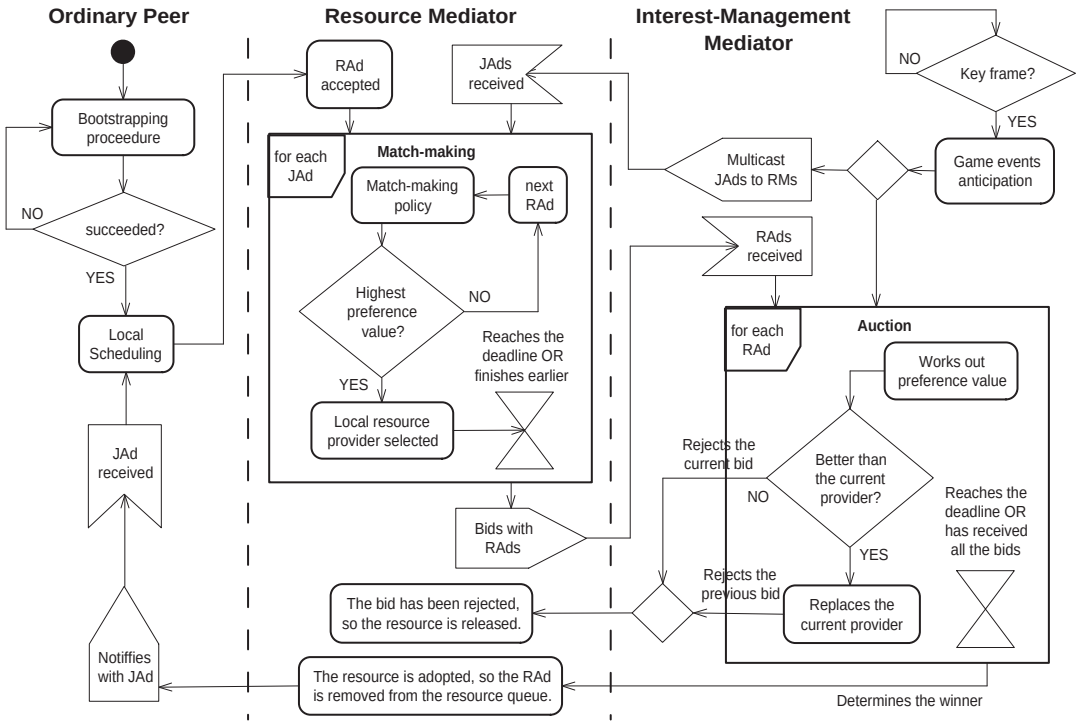

Figure 4.5: Interest-Management & Resource Mediator activity diagram

Figure 4.5 shows the resource discovery and scheduling activities carried out by the IMM and RMs. In short, the IMM maintains a global view of its zone and periodically anticipates forthcoming game events using a hybrid interest-management algorithm [168]. In case a NPC object is required according to game scenarios, the IMM distributes a real-time resource request (i.e. JAd) to all the RMs in the same zone. By receiving such a request, each RM selects a locally optimised resource provider and proposes it to the IMM as a "bid" (i.e. RAd). When the IMM has received all the bids, or the deadline for the task is sufficiently near, the IMM closes the "auction" and determines which resource provider hosts the NPC.

4.3.5 Synergy of the Super-Peers

Figure 4.6 shows collectively the activities, communication and collaboration among the ordinary and super-peers in a P2P MMOG that is designed according to the Mediator

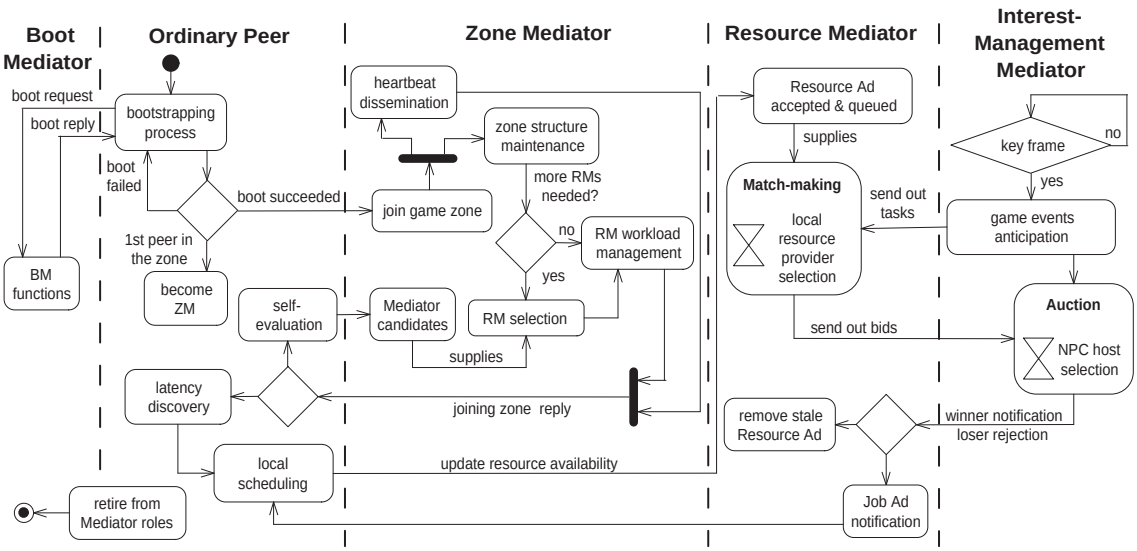

Figure 4.6: Activity diagram for the Mediator framework

framework. In summary:

- The BM is the rendezvous point of a game zone, at which an ordinary peer can find out the current ZM for the zone that it is about to join.
- The ZM constructs the super-peer infrastructure of a game zone, and roughly balances out the workload for multiple super-peer instances.
- The IMM carries out a fine-grained interest-management algorithm, anticipates forthcoming game events, and supplies the game world with NPC objects.
- The RMs manage zone members' available resources, and put forward appropriate candidates to the IMM to host NPC objects.
- The ordinary peers inform the IMM about their moving status, upload their resource availability information to the RMs, and contribute to the system by volunteering for super-peer jobs or super-peer backups at the ZM and hosting NPC objects for the public.

4.4 Super-Peer Dependability

At a macro-level, a P2P MMOG is more fault-tolerant than a C/S MMOG that relies on a centralised game server. As functional and administrative tasks are distributed to large numbers of application participants, the failure of some participants of a P2P MMOG will only affect a few other players in a short period of time. In contrast, if a game server failed, it would completely cut out the normal play of all the players supported by that server.

However, at a micro-level the robustness and availability of an individual super-peer is not comparable with a dedicated game server. So, it can be expected that various exceptions may often occur in a P2P MMOG, and a comprehensive P2P MMOG design framework must be able to handle such exceptions and enhance the dependability of its super-peers appropriately.

This section adopts a heuristic method to identify and address potential dependability issues in the Mediator framework by considering the following questions:

Q1. What kinds of exceptions may happen to each super-peer role? Here, a dependability exception refers to a super-peer failure that is caused either by various network, software and hardware problems, or the churn effect [83], i.e. constant joining and leaving of application participants. A presumption for the analysis is that all the application participants are honest and willing to cooperate for collective welfare. Situations, such as an unscrupulous super-peer providing false information to other peers that leads the system to an inconsistent state, are categorised as security issues, which should be addressed by cheating mitigation (Section 3.6) or reputation (Section 3.7) mechanisms. So, they are beyond the scope of this section.

Q2. How does a system detect such exceptions effectively and efficiently? This question is about the failure detection mechanism [43] that a system uses. An introduc-

tion to failure detection in asynchronous distributed systems is given in [138], where a simple yet widely employed approach is "heartbeat" [141]. This approach requires a node being monitored to send short "IAmAlive" heartbeat messages to its manager periodically to signify that the node is still operating. The node is said to "timeout" if the manager goes too long without observing a heartbeat message from it. In the case of a timeout, the manager could declare the node that timed-out had failed and then take whatever actions are necessary to recover the system from the exception [74]. As we shall see in the following sections, the Mediator framework uses a heartbeat as a primary way of detecting peer failures, and endeavours at avoiding the need for explicit "IAmAlive" messages so as to reduce communication overhead. Furthermore, an efficient membership management technology [65] will be discussed in Chapter 5.

Q3. How does a system recover from such exceptions automatically? A fault-tolerant system should be able to continue operating properly even when some of its components broke down. A typical way of achieving fault-tolerance is to provide multiple instances of the same system component and to switch over automatically to a redundant or standby component in case of a failure (a.k.a. failover). Redundant Array of Inexpensive Disks (RAID) [131] is a representative example. RAID1 enhances data reliability by replicating data to a redundant hard drive. Similarly, redundancy and replication are also helpful to improve the dependability of a P2P system. For example, in [165] the addresses of application participants which have higher computation powers and more available bandwidths are buffered in a "backup node queue". When a working super-peer is detected failed, one of these backup nodes will take over its job immediately. Moreover, a P2P system can also assign tasks or requests to a number of super-peers in parallel [132], so as to increase the probability of getting a correct or optimal result. As we shall see later on, the Mediator framework uses both of the two strategies flexibly.

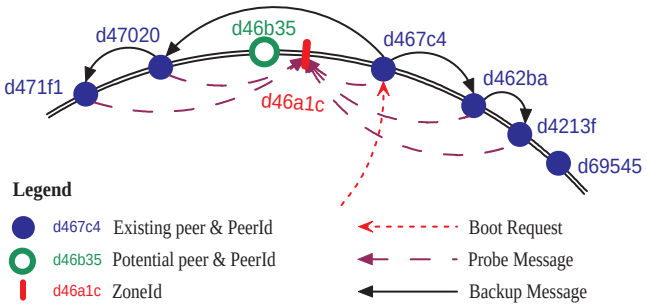

Figure 4.7: BM dependability enhancements

4.4.1 BM Dependability

To enhance the dependability of a BM, three exceptions must be handled properly:

- The leaving or failure of a working BM.
- The churn effect of a P2P overlay network.
- The unbalanced distribution of PeerIds.

1. The leaving or failure of a working BM Such exceptions will not affect the normal play of the corresponding game zone, because a BM is not involved in the game logic. However, it will result in the zone being closed to new game participants, because boot requests cannot be processed correctly. To prevent this from happening, a BM should replicate its knowledge of a game zone at its neighbouring peers.

Suppose that in Figure 4.7, BM `d467c4` is suddenly disconnected from the system due to a software crash. According to the current state of the overlay network, one of its immediate neighbours, i.e. peer `d47020` or `d462ba`, may become the new BM for zone `d46a1c`, depending on which PeerId is closer to the ZoneId. If these two peers were not aware of the status of zone `d46a1c`, when they receive a boot request from a newly joined peer, they would incorrectly infer that the peer is the first member of

that zone, and thus would promote the peer to a ZM. Consequently, the system will be put into an inconsistent state, which may take significant time and inconvenience to recover from. Instead, if BM d467c4 has replicated its knowledge of the current ZM at its immediate neighbours periodically, when peer d47020 or d462ba receives a boot request in an exceptional circumstance, they would be able to process the requests correctly and to step up as a BM substitute.

Furthermore, it is also possible for BM d467c4 to leave or fail with both of the immediate neighbours simultaneously. In this case, zone information is still lost, and the new BM that is two hops away from d467c4 could not process boot requests correctly. In practice, a working BM can recursively backup its information at $2\star K$ neighbours, so that only when all the $2\star K+1$ consecutive peers fail simultaneously, which is a low probability event, is the information lost. For example, when $K=2$, BM d467c4 replicates its information at peer d47020 and d462ba, which in turn replicate at peer d471f1 and d4213f. In fact, a structured P2P overlay can usually tolerate failures of about 16 consecutive peers, which means that when $K=8$, a BM can achieve equivalent dependability to the underlying overlay network, which is adequately robust.

2. The churn effect of a P2P overlay network A P2P MMOG is a dynamic system, as peers may leave and join the system at arbitrary times. The paragraphs above only discussed the departing of a working BM, whereas the arriving of new peers can also have an impact.

In Figure 4.7, if a new peer d46b35 joins the overlay network, its PeerId is closer to the ZoneId d46a1c than the current working BM d467c4. As a result, when some peer wants to join zone d46a1c, its boot request will be routed to peer d46b35 by the overlay network, rather than to the BM for that zone. However, peer d46b35 does not know anything about the zone, and thus will regard the sender of the request as the first member of the zone by mistake. To prevent this from happening, a working BM should detect the emergence of a nearby peer in terms of the Id space locality, and give up its role to that peer when necessary.

A simple way of detecting such situations is to ask a working BM to route small probe messages to the ZoneId periodically to simulate boot requests. Usually, such probe messages are delivered to the BM itself and discarded. However, when a new peer like d46b35 appears, the probe messages are delivered to the new peer. On the one hand, the new peer learns that there is an existing BM close to it, so it waits until the previous BM passes on necessary information, before starting to process boot requests. On the other hand, the previous BM detects the newly joined peer, and sends over its working information quickly.

Sometimes, only to ask a working BM to send out such probe messages is not sufficient. For example, if BM d467c4 has just failed before peer d46b35 shows up, the former would not detect the latter, and the latter would not fetch necessary information from peer d47020 or d462ba either. So, a more reliable approach is to ask both the working BM and its backup peers to issue probe messages periodically. In the case of a BM failure, the probe messages would be delivered either to one of the existing backup peers, or to a newly joined peer like d46b35, so that it is always guaranteed that a BM substitute is selected automatically in a timely fashion.

The last thing that needs to be tidied up is the backup peers that fall out of range. For example, when K=2, BM d467c4 replicates its data at four neighbouring peers, which are d47020, d471f1, d462ba and d4213f. As peer d46b35 joins, the previous BM becomes a backup peer, and the furthest backup peer d4213f falls out of range. So, when the new BM d46b35 receives a probe message from peer d4213f, it replies with an unemploy message that tells peer d4213f to stop probing.

3. The unbalanced distribution of PeerIds Section 4.3.1 noted that although the ZoneIds are evenly distributed across the Id space, unbalanced distribution of PeerIds may still result in a single peer being the BM for multiple game zones. Hence, application developers should take into consideration this potential issue and enable a BM module to handle boot requests for different zones, as well as to carry out related K

backup and probing procedures for those zones separately. As more peers join the system, such unbalanced distribution is likely to disappear, and the previous BM will have the opportunity to hand over the workload for some game zones to other newly joined peers.

In summary, a BM is subject to its own failure, the churn effect of the underlying overlay network, and occasionally, unbalanced distribution of PeerIds. To cope with these exceptions, data replication, redundant backup peers and failure detection mechanisms are needed. According to the characteristics of a BM, the ideal locations for data replication are its immediate neighbours on both sides. Also, a BM may require its immediate neighbours to replicate the data at more neighbouring peers recursively. As a result, up to $2 * K$ redundant backup peers can be recruited. Then, the working BM and its backup peers simulate boot requests together using periodical probing messages, so as to detect possible exceptions and to activate a BM substitute automatically.

So far, the BM activities depicted in Figure 4.3, including the processing of boot requests, the K backup and the periodical probing have been clarified. The last unclear bit of this diagram is a BM's monitoring of a corresponding ZM, which will be explained in the next section.

4.4.2 ZM Dependability

Compared to a BM, a ZM, IMM and RM are subject to fewer kinds of exceptions, because they are not selected according to the Id space locality and changes to the topology of the underlying P2P overlay network do not have an impact on their functioning. Hence, these super-peer roles are mainly subject to their own failures caused by network connection, software or hardware problems.

A ZM is responsible for zone structure maintenance, as described in Section 4.3.2. The services that a ZM provides are stateful, as it requires historical information about a game zone. Typically, a ZM maintains the following information:

- A manifest of all the zone members.
- A list of working super-peers, including one IMM and multiple RMs.
- A list of qualified super-peer backups.

When a ZM stops working, the game zone that is controlled by the ZM will no longer be responsive to changes to the zone structure, e.g. the leaving of a RM. However, if the zone structure remains the same, a ZM's failure will not affect the normal play of existing zone members. So, it is possible to reinforce a ZM's dependability with redundancy and replication, by replacing a failed ZM with a super-peer backup which has a valid replica of the ZM's information. Furthermore, in the worst case when the ZM and all its backup peers fail simultaneously, a "rebooting" of the entire game zone can be performed.

With reference to failure detection mechanisms, heartbeats seem to be an efficient way of telling a ZM's working status. A ZM needs to update its zone members periodically about the latest zone structure via an application-level multicast, as discussed in Section 4.3.2. These multicast messages can be used to replace explicit "IAmAlive" messages, and if the multicast stops for a sufficiently long time, it suggests that the ZM has ceased operation. Furthermore, because such heartbeats are sent to all the zone members, when they stop, every zone member is able to tell this and there are many different ways of initialising a ZM recovery process. For example, a P2P MMOG may entitle all the super-peer backups that have a valid replica of ZM information to negotiate and elect a ZM successor among themselves. The current design of the Mediator framework advocates consigning the monitoring and replacing of a ZM to the corresponding BM for that zone, due to the following considerations:

- A BM is a sufficiently dependable super-peer role. By choosing an appropriate K value, a BM can achieve an equivalent reliability to the underlying overlay network. Therefore, to ask the BM to monitor the ZM directly is a more reliable and straightforward solution than to ask another set of peers to monitor the ZM.

- When multiple peers are used to monitor the ZM, an extra negotiation protocol is required for a ZM successor to be elected. During the negotiation, many messages need to be exchanged, which induces unnecessary communication overheads and may delay the selection of a new ZM. Comparatively, it is easier and quicker to ask the BM to appoint a new ZM from super-peer backups directly.

- In practice, a BM can subscribe to the multicast group of its game zone to monitor the ZM's heartbeat. In case the heartbeat stops for a sufficiently long time, the BM would publish a "resuscitation" anycast [39] message within the group, which rapidly travels through every zone member, until it reaches the first super-peer backup that has a valid replica of ZM information. This backup peer is then selected to be the new ZM. So, the BM does not need to know how many ZM backups are available in the zone and which peers they are.

- BM's resuscitation can be repeated at fixed intervals until it works. However, if the working ZM and all its backup peers fail simultaneously, the ZM's information could be permanently lost, and the zone heartbeat could not be restored. After several resuscitation attempts, the BM will publish a reboot message in the multicast group, asking the zone members to bootstrap again. As a result, the first peer to rejoin the zone is promoted to be the first ZM, leaving a formal ZM to be selected later, and the whole super-peer infrastructure will be reestablished at last. This can be used as an ultimate recovering mechanism, which takes effect even when the super-peer infrastructure of a game zone is seriously damaged.

4.4.3 IMM, RM & NPC Host Dependabilities

The IMM, RM and NPC hosts are discussed together in this section, because their ways of reinforcing their dependabilities are quite similar. These special peers are all selected from resource-rich zone members. As a ZM maintains a manifest of all its zone members, it is convenient for the ZM to find out which of these peers are still working well

and which have left the zone or failed. The technology for the ZM to obtain such information efficiently is a membership-aware multicast that is discussed in Chapter 5.

Again, redundancy and replication are primary strategies for handling these peers' failures. For the IMM, its stateful working information that needs to be replicated at backup peers is a list that maps important NPC objects to their current hosts. As a game event anticipator, an IMM also works on other information, such as the positions, orientations and velocities of zone members. However, this information is transient, and thus does not need to be replicated. When the ZM detects the failure of the IMM, it selects a new one from the super-peer backups and informs the zone members about this change in the next heartbeat message. Accordingly, players in the game zone switch to update their moving information at the new IMM. However, during the handover time, affected players will not perceive other player avatars or NPCs that have entered their AOI recently. When the new IMM is up and the hybrid interest-management mechanism is restored, those players may see some avatars and NPCs appear around them suddenly. Such disturbed gaming experiences should be tolerable if they do not occur frequently.

For the RMs, they only work on transient information, which is the temporal resource availability of some zone members, so they do not need to carry out any replication. When an RM is detected to have failed, the ZM notifies its zone members in the next heartbeat, and affected zone members can switch to upload their resource availability at any other available RMs. Because multiple RMs coexist in one game zone and work in parallel, the failure of some RMs will not affect the functioning of a game zone. However, as the resources controlled by a failed RM can not be exploited in a short period of time, the overall resource quality attained by the system might become lower. For example, resource provider $Alice$ is able to host a NPC and to provide better game interactivity than resource provider Bob. However, because $Alice$'s RM happens to fail at the matchmaking time, the NPC task may be allocated to Bob. When some of the RMs are not working properly, the scheduling results are unlikely to be optimal.

For the NPC hosts, their importances are determined by the NPC objects that they are

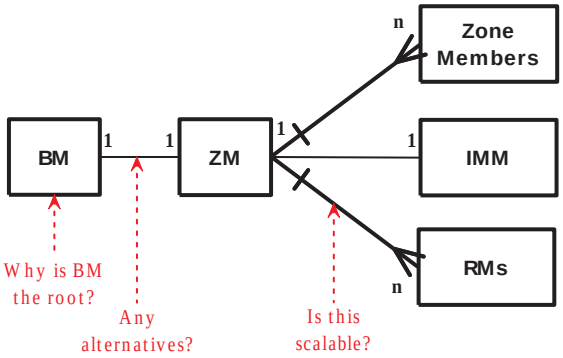

Figure 4.8: Hierarchical supervision relationships in a game zone

hosting. Some NPCs are relatively more important, such as a special character that drives a continuing storyline. When a game participant that is hosting an important NPC fails and is removed from zone membership in the ZM's heartbeat, the IMM will notice that and migrate this NPC to another capable resource provider. However, some other NPCs are less important, such as trivial monsters that combat with player characters. In this case, the IMM adopts a "let it fail" strategy and supplies the game world with more such NPCs according to game scenarios. Currently, the Mediator framework assumes that a NPC host does not need to replicate any data, because on the one hand the AI program and appearance of a NPC is part of the game software, and on the other hand, the stateful information required to initialise some important NPCs is usually part of the persistent game world and is stored and retrieved through a distributed persistency mechanism (Section 3.5). If a NPC host happens to fail in the middle of an interaction, the temporal state of the corresponding NPC could be lost. However, this should be tolerable if it does not occur frequently.

4.4.4 Hierarchical Supervision Architecture

In summary, the discussions from Section 4.4.1 to 4.4.3 result in a hierarchical supervision architecture as illustrated by Figure 4.8, in which:

- A BM monitors a ZM by listening to its heartbeats. When the heartbeat has

stopped for a sufficiently long time, the BM sends out a resuscitation message using an application-level anycast service provided by the underlying P2P overlay network to activate a ZM backup peer.

- The ZM in turn monitors a single IMM and multiple RMs. When any of these super-peers are detected to have failed, the ZM will replace them with capable super-peer backups, and notify the changes to its zone members via heartbeat messages.

- The ZM also monitors the presence of other zone members, including NPC hosts and ordinary players. In case a zone member is detected to have failed, the ZM would remove it from zone membership in its heartbeat. If the zone member was hosting an important NPC, the IMM would notice that and carry out necessary recovery behaviours.

To judge whether the design of the supervision architecture above is appropriate we need to reflect on the following points:

1. Why is the BM the highest level supervisor in this architecture? The Mediator framework comprises three conceptual layers as discussed in Section 4.2.1. It is important to notice that the lowest layer is a structured P2P overlay network, which provides the most fundamental services that support the functioning of a P2P MMOG, such as message routing and forwarding. However, an overlay infrastructure is not absolutely robust, as it is only able to sustain a certain level of churn. For example, a Pastry ring can tolerate failures of up to 16 consecutive peers [143], otherwise its routing algorithm will be broken. In this case, the ultimate goal of a supervision architecture is to help a P2P MMOG achieve an equivalent robustness to the underlying overlay infrastructure, and by tuning the K value, a BM is just able to meet this goal as discussed in Section 4.4.1. Therefore, it is reasonable to make the BM the highest level supervisor, as it is likely to be the last super-peer role that works correctly in a severe accident.

2. Are there any alternatives for monitoring a ZM? A short answer to this question is "yes", because a ZM's heartbeat messages are delivered to all its zone members, and once the heartbeat has stopped for a sufficiently long time, every zone member could tell that the ZM has ceased operation. As ZM backup peers can also tell this exception by themselves, it is possible for one of them to step up as a new ZM automatically. So, to ask the BM to monitor the ZM may be an easier and quicker solution as discussed in Section 4.4.2, but it is not the only possible solution. The most significant advantage of the design is that it makes a P2P MMOG as dependable as the underlying overlay network. Even if the working ZM and all of its backup peers failed simultaneously, the BM would still be able to recover the zone through a rebooting process. However, a disadvantage of the design is that to monitor a ZM's heartbeat will result in additional communication overhead, whereas a BM should have as little workload as possible as discussed in Section 4.3.1.

3. Is the ZM design scalable? It seems that the supervision workload imposed on a ZM is overly excessive, as it is required to monitor the working states of all its zone members. Actually, Figure 4.8 just depicts the logical supervision relationships among the super-peer roles, and in practice a ZM does not need to monitor so many peers by itself. Because a ZM disseminates heartbeat messages periodically to its zone members via application-level multicast, a multicast tree is established within each game zone. This communication infrastructure can be reused by an efficient membership management technology, which distributes a ZM's supervision responsibilities to large numbers of information forwarders in a multicast tree, so that not much workload is put on the ZM directly. This distinctive technology is called Membership-Aware Multicast with Bushiness Optimisation (MAMBO), whose design, implementation and evaluation are presented in Chapter 5.

4.5 Mediator Design Comparison

This section compares the Mediator framework with other existing P2P MMOG architectures, analyses their characteristics, and reflects on their relative advantages.

4.5.1 Mediator vs. Hybrid P2P MMOG Architectures

A crucial reason for a game server infrastructure being so expensive is that a substantial computation and communication workload is imposed on game servers. Therefore, a promising way of mitigating the game servers' workload is to distribute some of their functionalities to resource-rich application participants in a P2P fashion. Accordingly, a number of hybrid MMOG architectures have been proposed, e.g. [139], [108] and [91].

These architectures are referred to as "hybrid", because centralised game servers are still present in their design and usually play an important role. For example, in [139] a hybrid P2P MMOG initialises with a main server, which stores players' account information, maintains the global game state and supports players in joining and leaving the game. As the number of players increases, the main server delegates more and more of its role as game and communication manager to the connected player machines. Concretely, a hybrid MMOG architecture partitions its game world into multiple sub-spaces, and a single resource-rich participant machine (a.k.a. "region server" [139], "subserver" [108], or "zone owner" [91]) is selected by the main server to take charge of each sub-space. The main server delegates its responsibilities, such as communicating with game clients, processing game events, hosting NPC objects and updating game states, to these special peers. In other words, these peers serve as the authoritative game servers for corresponding sub-spaces.

Such hybrid architectures attempt to combine the advantages of C/S and P2P architectures. On the one hand, trusted game servers, e.g. a database server that stores the global game state, can be used to reinforce the security of a game. On the other hand, because a

set of participant machines are employed to regulate game servers' workload, a MMOG may scale up at a lower cost than using conventional C/S architectures. However, a significant limitation of these approaches is that their distribution of game servers' functionalities is coarse-grained, and thus they cannot make use of all the spare computing resources available in a P2P system efficiently. Though some resource-rich participant machines are recruited to help the main server, a larger proportion of participant machines are not utilised.

In contrast, the Mediator framework distributes a game server's functionalities to considerable numbers of participant machines in a fine-grained mode. A game zone comprises four major super-peer roles, each carrying out a subset of a game server's responsibilities, as discussed in Section 4.3. Peers may carry out self-evaluation and qualified peers may volunteer for a super-peer job or register as a super-peer backup. In addition, any application participants may contribute their spare resources by hosting NPC tasks for others. Hence the design of the Mediator framework offers more opportunities for the participants of a P2P MMOG to collaborate and brings the potential of a P2P system into full play. Furthermore, when large amounts of resource offers are available, resource heterogeneity can be exploited to provide better quality of service. For example, among multiple NPC host candidates, the one that has the shortest communication latency with target players can be selected to provide a smoother gaming experience, as discussed in Section 4.3.4. As long as a sufficient proportion of application participants are willing to cooperate, a P2P MMOG could be self-sufficient. The ultimate goal of the Mediator framework is to avoid the need for centralised game servers and to support a large scale P2P MMOG at very low, or even no cost.

Of course, it is also possible for the Mediator framework to adopt dedicated game servers, but the purpose of doing that is just to enhance system robustness and quality of service. For example, when many player avatars crowd in a popular area of a game world, the workload for the IMM in charge of that zone will become especially high. If the ZM could not find any competent resource providers that have adequate processing power and network bandwidth to serve the IMM role, the ZM would have to divide the

zone into four sub-zones, as discussed in Section 4.3.2. However, if a powerful server-peer is available in the system, it can take over the IMM job temporarily, until the IMM's workload returns to a normal level and can be handled by a player super-peer. Similarly, a server-peer can take over multiple super-peer jobs simultaneously in case various exceptions have happened, and the server-peer may give those jobs back to player super-peers whenever it is appropriate. In this way, an arbitrary number of server-peers can be introduced and removed from a P2P MMOG flexibly without affecting the functioning of the system. Such server-peers may be provided either by a MMOG service provider, or generous players who would like to dedicate their computers to a MMOG even when they are not playing the game.

4.5.2 Mediator vs. Fully Distributed P2P MMOG Architectures

A number of fully distributed P2P MMOG architectures also have been proposed in the literature, including [59, 86, 57, 79, 118]. The purpose of these architectures is similar to Mediator, and they all aim at supporting a MMOG using a pure P2P network without conventional game servers. Compared to these architectures, the Mediator design provides the following advantages:

1. Comprehensive Six representative fully distributed MMOG architectures are analysed in Section 3.8, and their ways of addressing the key design issues for P2P MMOGs are compared in Table 3.1. Among them, the Mediator framework is evaluated as *advanced*, because its design addresses the six key issues in an integrated system. Section 4.2.3 further elaborates on how each individual issue is addressed in the Mediator framework.

2. Robust A P2P MMOG relies on game participant machines to carry out various computational and administrative tasks. However, these user-supplied resources are less reliable than dedicated server resources, so fault-tolerance in face of node or link failure

is important [86]. Some related work, such as [59] and [57], does not discuss failure detection and recovery mechanisms, while some others resort to a simple redundancy strategy. For example, capable backup peers are used to replace failed region controllers in [79], or coordinators in [118], or arbitrators and aggregators in [86]. Unfortunately, the descriptions of these architectures do not explain how these backup peers are selected, in what way super-peer failures are detected effectively and efficiently, and when multiple backup peers are available how a super-peer successor is elected. In contrast, the Mediator framework presents a hierarchical supervision architecture to reinforce the dependability of all its super-peer roles. This supervision scheme allows a super-peer infrastructure to recover from various exceptions automatically, and to achieve a theoretical robustness that is equal to the underlying P2P overlay network. Furthermore, the Mediator framework can use a novel membership management mechanism (Chapter 5) to reduce the communication overheads incurred by the supervision scheme.

3. Appropriate Super-Peer Selection The Mediator framework selects super-peers according to participant machines' actual resource availability, and encourages resource-rich peers to volunteer for super-peer jobs using cooperative incentive mechanisms (Section 4.3). An exceptional case is that BM super-peers are selected using the Id space locality in a P2P overlay network so as to achieve self-organisation. There is no guarantee that a randomly selected super-peer like a BM can always provide adequate processing power and network bandwidth. Therefore, the workload assigned to a BM is lighter than other super-peer roles, and it is expected that any typical participant machine should be capable of performing a BM's service. In contrast, some related work, such as [118] and [79], selects super-peers in a random way, uses these super-peers as regional servers and imposes much workload on them. Such an approach to super-peer selection and load-balancing does not look appropriate.

4. Meeting Inherent Challenges for P2P Applications General P2P applications, including P2P MMOGs, are confronted with a few inherent challenges, as discussed in

Section 2.5.2. These challenges have been considered through out the design of the Mediator framework, and often played a decisive role. Firstly, most Internet users suffer from asymmetric network bandwidth. Specifically, the amount of downstream bandwidth is usually more than the amount of upstream bandwidth. However, the bandwidth requirement of a game server is quite the reverse, because one inbound game event may result in many outbound messages being reflected to the clients that are affected by the change of the game state. Hence in a P2P network few peers seem likely to have sufficient upstream bandwidth to provide a service that is similar to a game server. However, many hybrid [139, 108, 91] and fully distributed [79, 118] MMOG architectures require player super-peers to work as regional game servers. The practicability of these architectures is inevitably in doubt. The Mediator framework avoids this problem by breaking down a game server's functionalities and assigning them to different super-peers in a finer-grained way. On the one hand, time critical game events, such as the interaction among a NPC host and ordinary players, are exchanged via direct P2P communications. The consumption of upstream bandwidth at each peer is kept low. Furthermore, application-level multicast is employed when a super-peer needs to disseminate information to large numbers of receivers, e.g. via ZM's heartbeat messages. A multicast tree may lead to a higher end to end delay, but the messages that are disseminated in this way are not time critical, and thus such delay is usually tolerable.

Secondly, due to geographical distances and different Internet service provider's network configurations, the communication latency among game participants may vary. For example, let us suppose that players $Alice$, Bob and $Chris$ are in the UK, and player $David$ is in the USA. In this case, the communication latency between $Alice$ and Bob will be smaller than the latency between $Alice$ and $David$. Furthermore, the communication latency between $Alice$ and $Chris$ could be even smaller, if they used the same broadband service and their computers were on the same site of a metropolitan area network. Consequently, when $Alice$'s machine is selected to be a super-peer, it is not able to provide equal quality of service to all of its clients, such as Bob, $Chris$ and $David$, even though $Alice$'s machine has sufficient network bandwidth. Considering this fact,

the Mediator framework adopts a novel NPC host allocation mechanism that exploits peers' resource heterogeneity and allocates a NPC to the host that has the shortest communication latency with players in the vicinity of the NPC (Chapter 6). Comparatively, other P2P MMOG architectures that use region based or virtual distance based NPC host allocation mechanisms all suffer from various drawbacks, as discussed in Section 3.4.

Finally, both the Mediator framework and other P2P MMOG architectures require a minimum number of users to ensure the availability of player's account information and the global game state. Also, they all need to address the security issue, which has been a long term challenge for all kinds of online games. To date these issues have not been completely solved by any existing P2P MMOG architectures, and significant research efforts are being made into the investigation of useful technologies. For example, the PAST [144] distributed storage middleware discussed in Section 3.5 is a promising approach towards game state persistency for P2P MMOGs. PAST is built on the Pastry overlay infrastructure [143], which is also the substrate of the Mediator framework, as discussed in Section 4.2.1. So, it can be fairly convenient for Mediator to adopt a stabler and maturer version of PAST when it becomes available. Furthermore, real-time interactions among players and NPCs can be secured either by proactive cheating mitigation approaches like NEO [73], SEA [76] and EASES [42], or by reactive cheating mitigation approaches like Log Auditing [97], DaCAP [115] and FreeMMG [40], as discussed in Section 3.6. Compared to other P2P MMOG architectures, Mediator demonstrates better compatibility with the latest game state persistency and cheating mitigation technologies, and has better potential for addressing these issues more fully in the near future.

4.6 Discussion

Mediator is a novel and broadly comprehensive design framework for P2P MMOGs. Compared to other hybrid MMOG architectures, the Mediator framework breaks down a game server's functionalities and distributes them to large numbers of game participant

machines in a more fine-grained manner, so as to achieve more efficient resource utilisation. While hybrid MMOG architectures aim at reducing the costs of a conventional game server infrastructure, the Mediator framework aims at supporting a large scale P2P MMOG at lower or even no cost.

Furthermore, compared to other fully distributed MMOG architectures, Mediator's advantages lie in that it addresses the six key design issues for P2P MMOGs in an integrated system; provides a hierarchical supervision architecture and an efficient membership management mechanism to endow a P2P MMOG with the same level of robustness as a P2P overlay network with limited maintenance overhead; selects super-peers and NPC hosts appropriately according to participant machines' actual resource availability; and takes into consideration more inherent challenges for general P2P applications, such as asymmetric network bandwidth, heterogeneous resources, and compatibility with the latest persistency and security mechanisms. Hence the Mediator framework has better potential to adapt MMOGs from conventional C/S architectures to P2P architectures.

— A great part to the information I have was acquired by looking up something and finding something else on the way.

Adams Franklin

Chapter 5

Membership-Aware Multicast with Bushiness Optimisation

5.1 Introduction

Multicast is the delivery of information to multiple hosts which have joined an appropriate communication group. It is an important component of many network applications, which require efficient one-to-many and many-to-many communication infrastructures. Examples include:

- Multimedia streaming, in which a single media source broadcasts to a large number of receivers, e.g. audio webcasting, live sports and TV program streaming.
- Web conferencing, in which multiple parties share data in real-time such as audio, video, presentation slides, text chat and whiteboards.
- Multi-player online gaming, in which participants may frequently generate events and messages that affect the in-game state of other participants.

- Generic publish-subscribe and event notification services, in which subscribers subscribe to and receive from publishers events that they are interested in.

In the literature, IP multicast [56] has been proposed as a mechanism for sending data to a group of recipients efficiently. However, due to a number of technological, practical, and business obstacles [58], IP multicast is not widely used. The alternative Application-Level Multicast (ALM) has been proposed to support similar functionalities, but as an application service instead of a network service [63]. A survey of existing ALM systems is given in [84], which identifies their characteristics, compares their protocols, and analyzes current trends.

Different P2P applications impose a range of requirements on membership management. Some applications do not care whether a participant is still present in the system, and a simple "let it fail" strategy can be used to handle participants' failures. However, in some other applications, the arrival and departure of application participants need to be detected and managed within a short time period (Section 5.2). This has sparked this research's interest in Membership-Aware Multicast (MAM).

This chapter presents the conceptual design of MAM and its extension for load-balancing purposes, namely Bushiness Optimisation (BO) (Section 5.3). MAM reuses the communication infrastructure of a tree-based ALM system to track group membership, i.e. to record when peers join or leave the group. Currently, MAM and MAMBO are implemented as policy plug-ins for Scribe [38] which is available with Pastry [143] (Section 5.4). A demonstration application, Peer-to-Peer Online Market Place (POMP), has been developed, with which the effectiveness, scalability and communication overheads of MAMBO have been measured and compared with a conventional service-provider architecture. (Section 5.5).

The experimental results in Section 5.5 show that MAM and MAMBO quickly detect, and effectively manage, the arrival and departure of peers. In addition, MAMBO delivers a number of other advantages. The bushiness optimisation limits the overhead placed on

any peer, preserving the scalability of the ALM system. The implementation as Scribe policy plug-ins demonstrates that MAMBO can readily be overlapped on a tree-based ALM, and requires few changes to the underlying technology.

5.2 Motivation

A framework for classifying P2P applications is given in [99], together with a range of examples like file sharing and instant messengers. The framework indicates that different P2P applications have varying requirements on membership management. For example, a P2P file sharing application does not need to tackle this issue, because from a downloader's perspective, the leaving of an uploader may only affect the quality of service, e.g. resulting in a slower download speed, and from an uploader's perspective, the leaving of a downloader can be ignored, as the uploader can switch to serve other downloaders. In other words, a P2P file sharing application only needs to provide best-effort services.

However, this is not the case in some other P2P applications, which have much higher requirements regarding membership management. This section identifies several such scenarios in order to motivate this research's investigation of membership-aware multicast.

5.2.1 Collaborative & Distributed Computing

P2P applications for collaborative and distributed computing employ application participants' computing resources such as CPU cycles, memory, and storage capacity to perform critical service functions in a decentralized manner. For example, the Mediator framework described in Chapter 4 supports a P2P MMOG by a pooling of game players' computing assets. Consequently, a substantial proportion of application participants are responsible for various computational or administrative tasks, and thus the loss of any

participant needs to be detected and handled in due course, otherwise it may damage the integrity and consistency of the system. Therefore, Mediator aims to provide dependability using a hierarchical supervision architecture, as discussed in Section 4.4. In particular, a Zone Mediator is both responsible for monitoring the presence of its zone members and disseminating periodic heartbeat messages to them via an application-level multicast. In this case, if the underlying ALM infrastructure used by the ZM comes with a membership management capability, the ZM would be able to find out which zone members are subscribing to its multicast group. As a result, there would be no need for the ZM to monitor the zone members by itself, and considerable communication overhead could be avoided.

5.2.2 P2P Incentive Mechanisms

P2P applications are by nature voluntary resource sharing systems, in which there is often a tension between individual concerns and collective welfare. As the benefits of these systems are rooted in cooperation, they are inherently vulnerable to non-cooperative behaviour, and it is especially necessary for such systems to be designed so that participants are induced to be cooperative. The mechanisms that are embedded in a P2P system for this purpose are called incentive mechanisms [172], and are crucial for maintaining the viability of a P2P application.

Event dissemination and notification are also common functionalities that are required by substantial numbers of P2P applications. Hence, an ALM infrastructure that supports membership management functionalities, such as tracking a peer's historical activities in a system [77], would be able to satisfy the communication and incentive requirements of a P2P application at the same time. The POMP application discussed in Section 5.4 demonstrates the advantages delivered by membership-aware multicast.

5.2.3 P2P Multimedia Streaming

P2P multimedia streaming is another well-known application type that relies on application-level multicast. Some P2P streaming applications, such as live sports and television broadcasting, happen in real-time and require the subscribers to be tightly synchronised with each other [170, 48]. However, some other applications, such as Video on Demand, store existing multimedia contents as a P2P file sharing application, and allow users to select and watch videos and clips over a network [82, 44]. Such applications could be made stateful, which means that an information forwarder monitors its subscribers' states, and in cases where some of them experience temporary network or software failure, the forwarder will cache the contents for them, rather than cut off the streaming. So, when the subscriber recovers from the failure, it may reconnect to all the previous forwarders and resume the streaming at the point of disconnection, instead of restarting the clip again from the beginning. It would be fairly convenient to fulfil such application requirements with a membership-aware multicast infrastructure.

5.3 Design

5.3.1 MAM Design

The key idea in MAM is to overlap an application-level multicast tree with a supervision tree that monitors and organises its membership, as depicted in Figure 5.1. Its main novelty is to reuse a communication infrastructure for a secondary purpose, namely membership management. In Figure 5.1, from the left to the right are:

ALM Tree Existing ALM systems can be classified into at least two broad categories: a gossip-based approach and a tree-based approach. A gossip-based approach, e.g. Chainsaw [130], Bar-Gossip [110] and MSCAMP [111], uses an epidemic model, in

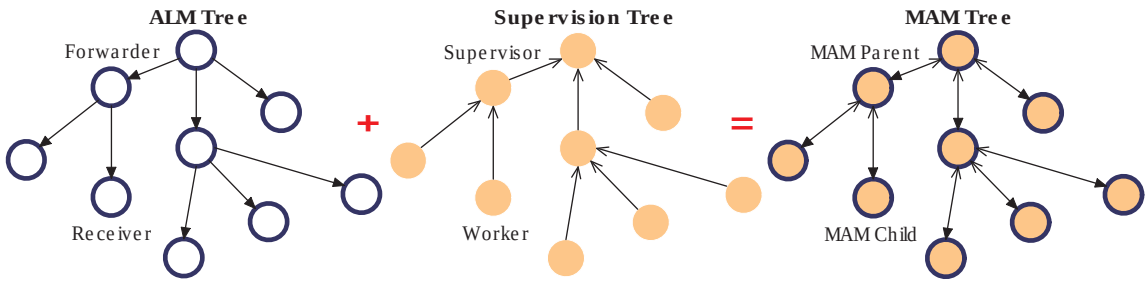

Figure 5.1: Conceptual design of Membership-Aware Multicast

which each application participant forwards information to a number of random "neighbouring" participants. Because such information dissemination is stochastic, there is not a stable communication topology in a gossip-based approach. In contrast, a tree-based approach, e.g. Chunkyspread [157], SMO [153] and Scribe [38], establishes a relatively stable multicast tree infrastructure among its participants, in which receiver peers keep subscribing to the same forwarder peers. The tree creation algorithm used by Scribe to do this was outlined in Section 3.3.2.

Supervision Tree The supervision tree concept was formulated by Joe Armstrong in his PhD thesis on the Erlang programming language [22]. A supervision tree is based on the idea of Workers and Supervisors. Workers are processes which do the actual work, and supervisors are processes which monitor the behaviour of workers. A supervisor is entitled to stop and restart a worker if something goes wrong with it.

MAM Tree The MAM tree combines a ALM tree and a supervision tree by mapping the logical supervisor-worker relationship to the existing information forwarder-receiver relationship in a communication infrastructure. In other words, MAM endows the forwarder peers with supervision responsibilities, so that each forwarder may monitor its own children' activities and carry out various management behaviours according to specific application requirements.

The most significant advantage of MAM is to provide a convenient way of membership-

management while reinforcing the dependability of P2P applications. Without MAM, a P2P application either requires a centralised service provider to monitor the presence and working state of its participants, or has to construct and maintain a separate hierarchical supervision infrastructure, which may result in unnecessary communication and computation overheads. Furthermore, because a MAM tree reuses an ALM tree, it automatically inherits all the advantages provided by the underlying communication infrastructure, such as being self-organising, scalable and fault-tolerant.

However, the main limitation of MAM is that it only applies to a P2P application which already employs a tree-based application-level multicast. Otherwise, there would not be an appropriate environment to deploy MAM. Currently, three application types that may exploit MAM are identified in Section 5.2. One avenue of future work would be to investigate MAM's potential use in further distributed applications.

5.3.2 MAMBO Design

The key idea in MAM with Bushiness Optimisation is to limit the number of children a MAM parent peer will accept. Because MAM directly reuses a tree that is created by an ALM system, the shape of a MAM tree may vary. Concretely, on a MAM tree, it is possible for some parent peers to have many more children than others and hence incur high overhead. Experimental evidence shows that a small number of MAM parent peers can experience very high workloads, e.g. the workloads for some parent peers in a 1000-peer tree are hundreds of times greater than for an idle parent peer (see Figure 5.13, Section 5.5.5). To address this, a bushiness optimisation mechanism has been introduced.

Figure 5.2 compares the shape of a MAM and a MAMBO tree with at most 3 children. The MAMBO tree is more balanced than the MAM tree, and hence the workload is more fairly distributed. However, a tradeoff is that the MAMBO tree may be deeper and induce greater end-to-end communication latency, e.g. the same 16-element MAMBO tree has a maximum depth of 4, whereas the MAM tree depth is 3.

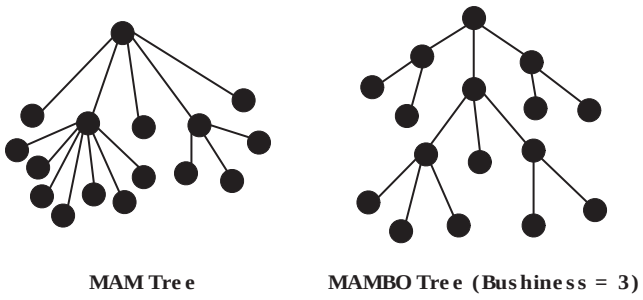

Figure 5.2: Contrasting MAM & MAMBO trees

The experimental results in Section 5.5.5 show that MAMBO is an effective load-balancing mechanism and enhances the scalability of a P2P application. However, it may also increase the probability of simultaneous child/parent failures. This negative side effect is discussed further in Section 5.5.6.

5.4 Implementation

5.4.1 MAMBO Prototype

Currently, a prototype of MAMBO has been implemented on top of the Scribe application-level multicast middleware [38] provided with Pastry [143]. MAM is implemented as a maintenance policy plug-in, and BO as a tree construction policy plug-in. Both of the policies only cause a few changes to the underlying technology.

MAM Maintenance Policy

The MAM maintenance policy establishes a supervision relationship between each MAM parent peer and its child peers using a heartbeat mechanism [141], as discussed in Section 4.4. On the one hand, a child peer periodically sends an "IAmAlive" (IAA) message to its parent, and on the other hand, a parent peer expects such IAAs to arrive on time,

```
//Maintenance_Interval is configurable
if(t >= Maintenance_Interval){
    //A node may subscribe to multiple groups
     for(each groupId)
        //Line-5.3a
         resubscribe(groupId);
}
```

Figure 5.3: Pseudo-code for MAM child peers

and when a child fails to do this, the parent will actively check the child's liveness. This supervision protocol can be expressed by the pseudo-code in Figures 5.3 and 5.4.

In Figure 5.3, the $MaintenanceInterval$ is a configurable parameter. A shorter interval can make a system to be more sensitive to membership updates, whereas it can also result in more communication overhead. Furthermore, because an application participant may subscribe to multiple multicast groups simultaneously, the participant may need to upload IAA messages to more than one parent peers.

Most importantly, $Line-5.3a$ indicates that a child peer sends IAA messages by resubscribing to a multicast group, instead of delivering the messages to its previous parent peer explicitly. This is because of Scribe's reverse path forwarding multicast tree construction algorithm discussed in Section 3.3.2. In Scribe, a subscribe request is routed by the underlying Pastry overlay from the subscriber to the root for the multicast group. Along the route, the request may terminate at any intermediary peer, which is already a part of the multicast tree, before arriving at the root. The intermediary peer then adopts the subscriber as one of its children and forwards any multicast messages to the subscriber. Therefore, if a peer's parent is still present in the system, when the peer resubscribes, its previous parent will receive and recognise the request as an IAA message. Otherwise, if the peer's parent has left the system or failed silently, by resubscribing, the peer will automatically attach itself to some other parent peer in the multicast tree. In this case, the MAM maintenance policy guarantees that a child peer is always well subscribed to the multicast groups that it is interested in, as well as enables a parent peer to monitor the presence and working state of its child peers. By efficiently reusing the IAA messages, MAM fulfils these two requirements at the same time with minimal

```
if(t >= Maintenance_Interval){
    //A node may have children in multiple groups
     for(each groupId){
        //Line-5.4a
        if(is root for this group)
             doBackup(leafset);
         for(each child nodeId){
             if(no IAA in the last Maintenance_Interval)
              //Line-5.4b
                 checkLiveness(child nodeId);
         }
    }
}
```

Figure 5.4: Pseudo-code for MAM parent peers

communication overheads.

$Line-5.4a$ in Figure 5.4 shows that unlike common interior peers, the root for a multicast tree does not have any parent supervisor, so the root has to backup the knowledge of its own children to some other nodes, in case it fails. This strategy is quite similar to the dependability reinforcement for BMs in the Mediator framework (Section 4.4.1), as a BM is the peer whose $PeerId$ is closest to a $ZoneId$, and a root is the peer whose $PeerId$ is closest to a multicast $GroupId$.

$Line-5.4b$ in Figure 5.4 shows that a parent peer would actively check the liveness of a potentially missing child, if the child fails to send an IAA message in the last maintenance interval. This is considered as a non-routine supplement to the IAA approach, and application developers may configure their MAM maintenance policies to ask a parent peer either to carry out such active checking, or to infer a child's failure passively when the child fails to report IAA messages in K continuous maintenance intervals.

Also, application developers can customise the structure and contents of an IAA message according to their needs. Typically, an IAA message should carry necessary information as evidence of the correct functioning of a child peer. However, in order to reduce communication overheads, as well as the workload for parent peers, the size of an IAA message should be kept small.

Finally, once a parent peer detects the departure or failure of a child peer, it should be

possible for the parent peer to handle this exception by itself, or to report it to other higher level supervisors. Currently, the MAM maintenance policy supports adjunctive decision-making policies, according to which a parent peer can solve problems that are within its authority locally and pass more significant problems on to appropriate handlers using an observer/observable pattern.

BO Tree Construction Policy

The BO tree construction policy entitles an intermediary peer to reject adopting a new child when it does not have enough communication bandwidth, or computation power, to forward content to, and monitor the child. Instead, the intermediary peer will pass the subscribe request on to another peer within that group as an anycast message. Anycast [39] is a Scribe service which allows a peer to send a message to a nearby member of a group, and guarantees that the message travels through the whole group, passing each member just once, until some member accepts the message explicitly. It is used for delivering a resuscitation message to a potential ZM backup peer in Section 4.4.2. By passing on a subscribe request as an anycast, a fully subscribed parent peer inquires about possible attachment points in the multicast tree on behalf of the subscriber.

The current tree construction policy enables a peer to define a maximum number of children that it accepts, i.e. its bushiness, and when this number is reached, the peer can determine in which way new subscribe requests are passed on as anycast messages in a multicast group, e.g. breadth first or depth first.

5.4.2 Alternative Service-Provider architecture

As a basis for comparison, an alternative service-provider membership management architecture is also implemented. In this architecture, the root peer for a Scribe multicast tree is assumed to be a dedicated server that monitors the presence and working state

```
if(t >= Maintenance_Interval){
    //Line-5.5a
     if(no comm in the last Maintenance_Interval)
         send(IAA);
}
```

Figure 5.5: Pseudo-code for service consumers

```
if(t >= Maintenance_Interval){
    for(each service consumer nodeId){
       //Line-5.6a
        if(no comm in the last Maintenance_Interval)
            checkLiveness(service consumer nodeId);
    }
}
```

Figure 5.6: Pseudo-code for the service provider

of all the application participants directly. Its supervision protocol is depicted by the pseudo-code in Figures 5.5 and 5.6.

At the first glance, this supervision protocol is quite similar to MAM's maintenance policy, because service consumers also periodically upload IAA messages to the service provider, and the latter expects the IAAs to arrive on time and actively checks the liveness of service consumers that fail to report on time.

However, in fact there is a significant difference between the service-provider architecture and MAM. In Scribe, when a participant of a multicast group wants to disseminate its information to other participants, it sends the information first to the root of the multicast tree, which in turn forwards the information to forwarder and receiver peers recursively. In this case, any communication between a service consumer and the service provider can be used as an optimisation to avoid the need for explicit IAA messages. This is named the Communication Overhead Reduction (COR) effect. So, in Figure 5.5 $Line-5.5a$ shows that a service consumer only needs to send out an IAA message when no other communications were made to the service provider in the last maintenance interval, and in Figure 5.6 $Line-5.6a$ also shows that the service provider is able to infer the presence and working state of a service consumer from ordinary messages.

In contrast, a COR effect cannot be exploited in MAM, because there is only unidirec-

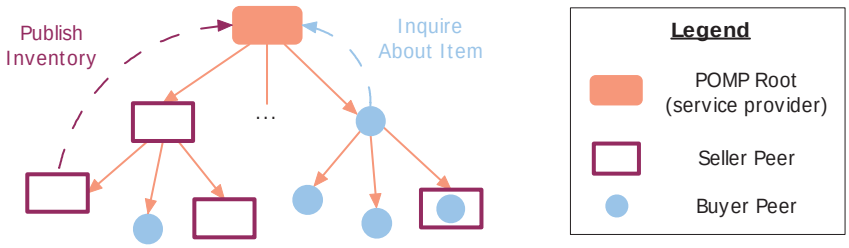

Figure 5.7: POMP system model

tional communication between a parent peer and its child peers, i.e. the parent forwards information to its children. Hence, there is no opportunity for a child peer to replace explicit IAA messages with other communication. The COR effect is quantified in Section 5.5.4.

5.4.3 POMP Demonstration Application

To measure the effectiveness, scalability and communication overheads of MAMBO, as well as to compare MAMBO to a conventional service-provider architecture, a POMP demonstration application was implemented.

Figure 5.7 depicts the system model of POMP, which consists of a service provider and a group of users, who can be buyers, or sellers, or both. The architecture of POMP is similar to PeerMart [81], an auction-based P2P market. POMP relies on an application-level multicast tree (a.k.a. POMP channel), which is rooted at the service provider to disseminate information among its participants. Sellers subscribe to the channel to publish their inventories (i.e. items for sale) periodically, and buyers subscribe to the channel to inquire about certain items that they are interested in. Each published inventory is cached for a certain period of time by all online participants that have received it, so it is possible for a buyer to look up recent available items using P2P content discovery [78].

According to the information dissemination mechanism of a tree-based ALM, both sellers and buyers have to send their inventory and inquiry messages to the root for the

Parameter	Value
Maintenance Interval	60 (seconds)
Length of a Liveness Checking	120 (seconds)
Permissible Billing Error	180 (seconds)
Length of an Experiment	100 (minutes)

Table 5.1: POMP configuration parameters & values

POMP channel, which in turn forwards them to other application participants recursively. In this case, it is convenient for the service provider to reinforce Authentication, Authorization and Accounting (A^3) [55] for all application participants. For example, the service provider can bill the participants in terms of virtual credits according to the number of messages that they have published, and intercept the messages from participants who can not afford the payment.

However, to establish a complete incentive model and to guarantee the viability of the application, the service provider may also reward the participants according to the time that they stay online, facilitating other people' inquiries. For this purpose, the service provider needs to record the time when a participant joins and leaves the application. It can either monitor all the participants directly using the mechanism discussed in Section 5.4.2, or delegate supervision responsibilities to intermediary participants using MAMBO as discussed in Section 5.4.1. By comparing the two different approaches, the advantages and disadvantages of MAMBO are investigated in the next section.

5.5 Evaluation

5.5.1 Experimental Setup

The experiments were carried out with the Direct network simulator provided by FreePastry, a Java-based open source implementation of the Pastry [143] structured P2P over-

lay infrastructure. The Direct simulator is a discrete event simulator, which can be executed either with a system clock, or with a virtual clock that runs much faster than real-time. The simulator supports a variety of network topologies, and a topology descriptor file created by GT-ITM [171] is used in the experiments to simulate Internet scale communication latencies among the application participants.

Furthermore, both MAMBO and the conventional service-provider architecture are implemented over Scribe, an ALM middleware built on FreePastry outlined as discussed in Section 5.4. In this case, the POMP demonstration application was also implemented in Java, and the experiments were performed in a single JVM hosted by a physical machine with 2.4GHz Intel Dual Core CPU and 2GB of main memory. Such a hardware platform is capable of running around one thousand POMP peers.

In each experiment, a number of simulated POMP peers boot into the same overlay network, select their roles as sellers or buyers randomly, and communicate with each other through the Direct simulator. Table 5.1 lists the parameters and their values used to set up the POMP application. The *Maintenance Interval* parameter discussed in Section 5.4 is set to 60 seconds, and parent peers are required to check the liveness of a child actively, if the child failed to upload an IAA message in the last maintenance interval. Such a checking process may take up to 120 seconds to complete, so normally a parent peer can discover the leaving or failure of its children within 180 seconds, which is referred to as the permissible billing error. This error range applies to both MAMBO and the service-provider architecture.

Each experiment simulates a POMP application session of 100 minutes, and is repeated 5 times. In order to speed the experiments up, the Direct simulator is configured with a virtual clock running at ten times the real-time rate. As a result, it is unfeasible to model a POMP peer using an individual thread which is not synchronized with the virtual clock. Instead, a POMP peer is modeled as an event-driven stub that schedules activities, e.g. publishing inventory/inquiry messages, joining, or leaving the application, on the simulator thread, and takes action according to the simulator's notifications. The controlling

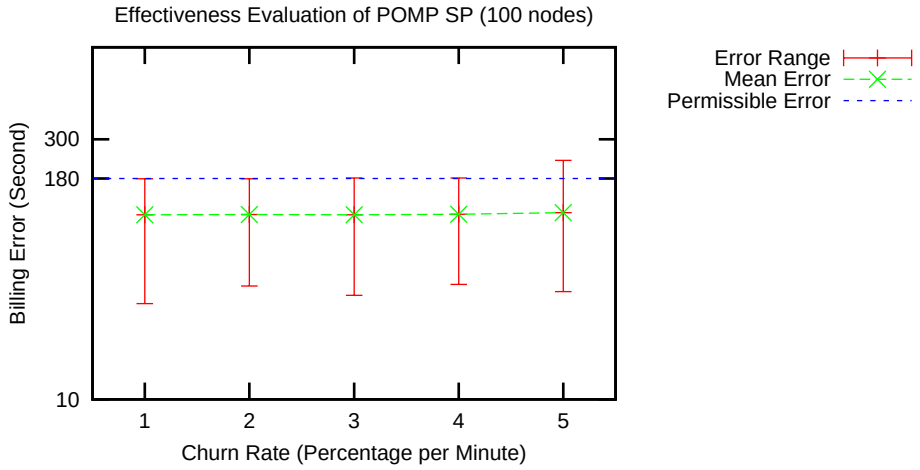

Figure 5.8: Effectiveness of the conventional service-provider architecture

of application churn rate (Section 5.5.2) and peers' talkativeness (Section 5.5.4) is also achieved via the scheduling of these activities.

5.5.2 *Effectiveness*

This experiment validates and compares the effectiveness of MAM, MAMBO(Bushiness=10) and the service-provider architecture by simulating five different churn rates with 100 POMP peers. The performance metric is their mean billing errors, which should be smaller than 180 seconds as discussed in the previous section. With the help of MAM or MAMBO, the service provider delegates supervision responsibilities to parent peers and registers itself as an exception observer. Each parent peer records the time when its child peers join and leave the application, and reports corresponding results to the service provider. On the other hand, each individual peer also records the time period that it has subscribed to the application. So, after the experiment, the billing error can be determined by comparing the service provider's estimations against each peer's own accurate record.

Furthermore, the churn rate in this experiment is expressed by the mean session length of the application participants [83]. For example, if the mean session length is 20 minutes,

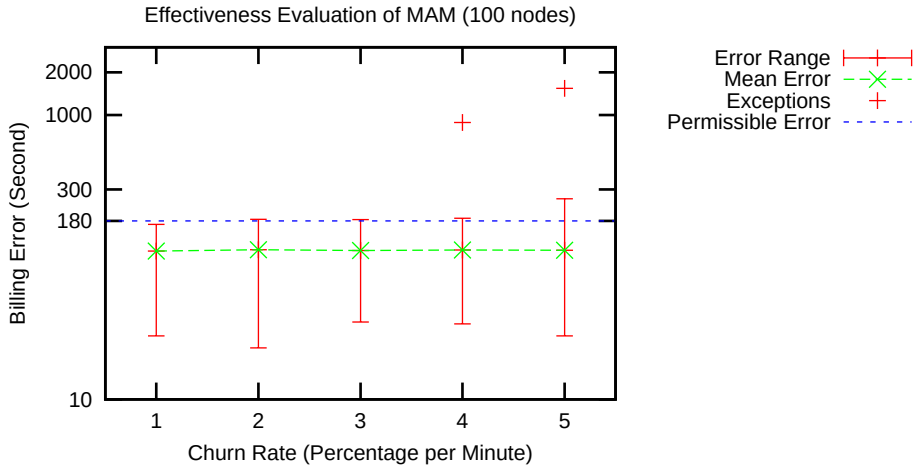

Figure 5.9: Effectiveness of MAM

it means that in every minute $\frac{1}{20}$th of the online population will leave the application. In the case where the mean online population is 100 peers, in every minute, there will be $\frac{100}{20} = 5$ peers leaving the application, resulting in a churn rate of 5% of the total population per minute. In this experiment the mean online population is kept steady at 100 peers, hence when 5% of the peers leave the application, another 5% new peers will join. The effectiveness of the service-provider architecture (Figure 5.8), MAM (Figure 5.9) and MAMBO (Figure 5.10) is measured under five churn rates that range from 1% to 5%.

Figure 5.8, 5.9 and 5.10 show that the three approaches are all able to provide an acceptable mean billing error below 180 seconds. Therefore, they are all considered to be effective on membership management. We make the following additional observations from the figures:

Maximal Billing Errors In most circumstances, the maximal billing error caused by the three approaches is within the 180 seconds threshold as expected. However, this is violated sometimes, e.g. for the service-provider architecture, under a churn rate of 5% the maximal billing error exceeds such a threshold. A conjecture with regard to this phenomenon is that a higher churn rate may lead to more workload on the service provider,

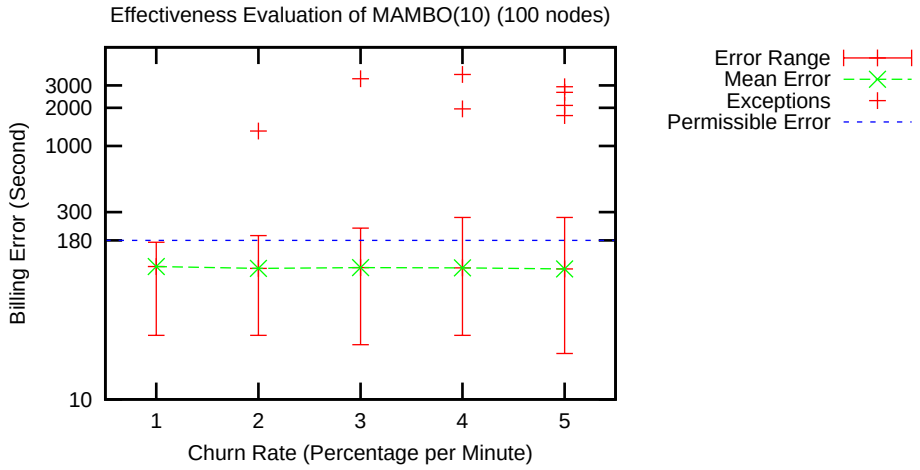

Figure 5.10: Effectiveness of MAMBO(10)

as it is monitoring all the application participants directly by itself. When a sufficient number of participants leave the application simultaneously, the service provider would have to check their liveness and thus form a performance bottleneck.

The conjecture cannot account for two important facts. On the one hand, the experiments are carried out on a single physical machine, and all communications are simulated. So, it is unlikely that the service provider runs out of processing power or network bandwidth. On the other hand, the same effect also happens to MAM and MAMBO, which actually distribute the supervision workload to multiple parent peers. MAMBO especially puts a limitation on the number of children that a parent peer may have, but its maximal billing error exceeds the threshold even earlier at a 2% churn rate.

Further investigation finally revealed the real reason for this issue. A high churn rate not only affects the application level, but also has an significant impact on the underlying Pastry routing algorithm. When an existing peer leaves the system and a new peer joins, other peers that have been present in the system will need to update their leaf sets in order to maintain a correct routing table. To do that, many messages need to be exchanged, and some complex computations need to be carried out. Because low level maintenance messages take a higher priority than common application level messages, the simulator will facilitate the delivery of the former, even though this may delay the

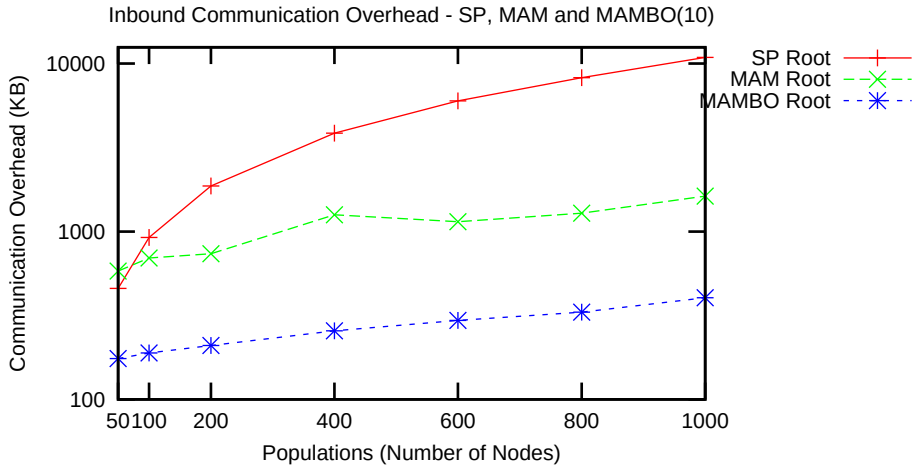

Figure 5.11: Scalability comparison

latter. The situation in MAMBO is worse, because MAMBO often needs an anycast message to be sent through a Scribe multicast tree until a parent peer that can accept a new child is found. Anycast messages also take higher priority than application level messages, so the delay occurs even earlier when there is only a 2% churn rate. We regard this issue as a drawback of carrying out experiments with a network simulator, and consider an implementation to be effective if its mean billing error satisfies the 180 seconds threshold.

Billing Exceptions In Figures 5.9 and 5.10 there are several exceptional results marked by red crosses. These billing errors are too long to be caused by a simulator delay, and indicate that the leaving of some peers was never detected until the end of an experiment. Such exceptional cases only happen to MAM and MAMBO, and it seems that MAMBO suffers the problem more often than MAM.

A careful analysis of experiment logs reflects that the reason for these outstanding errors is the simultaneous leaving of a parent peer and some of its children. In the service-provider architecture, because a dedicated service provider is always available, the leaving of any application participant can be detected. However, MAM and MAMBO distribute the supervision work to intermediary parent peers, which may leave the appli-

cation at an arbitrary time as well. Normally, when a parent peer has left, its children will automatically resubscribe to a new parent peer in the next maintenance interval by routing an IAA message. But, if some of these children happen to leave the application before they find a new parent, they would disappear from the application silently.

The inability to detect simultaneous leaving/failures of a parent peer and its children is regarded as a significant limitation of the current design of MAM and MAMBO, and is discussed further in Section 5.5.6.

5.5.3 Scalability

This experiment compares the scalability of MAM, MAMBO(Bushiness=10) and the service-provider architecture by increasing the online population gradually from 50 to 1000 peers. Figure 5.11 shows the change of inbound communication overheads imposed on the root for a multicast tree. This measurement can reflect the scalability of the three different approaches, because of the following reasons:

Inbound Communication Overhead The inbound overhead is selected as a performance metric instead of outbound overhead, because supervisor peers in all the three approaches mainly wait for IAA messages to arrive from their children. Outbound communication only happens when a supervisor peer needs to check a child peer's liveness actively, and to report an exception to higher level supervisors when it is necessary. Comparatively, such outbound overhead is much less than the inbound overhead for routine maintenance purposes.

The Root Peer In all the three approaches, the root for a multicast tree always tends to have more overhead than other supervisor peers, and thus it is more likely to become a performance bottleneck. On the one hand, the service-provider architecture requires the root peer, i.e. the service provider itself, to monitor all the application participants,

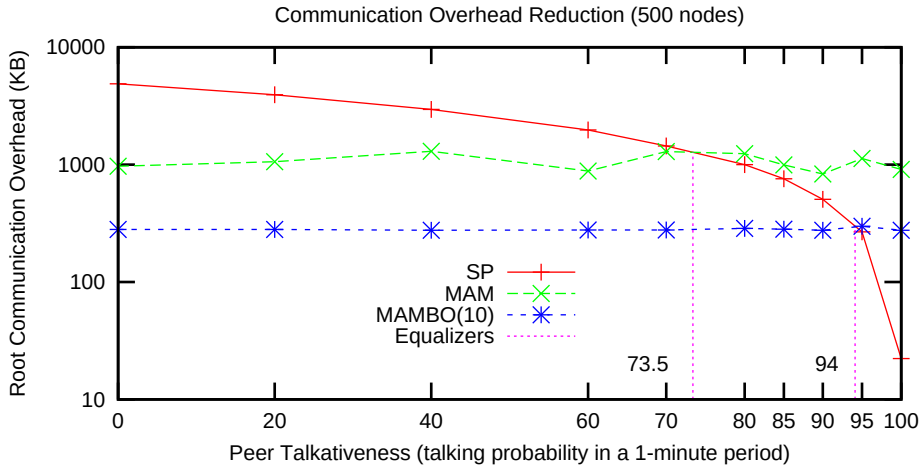

Figure 5.12: Communication overhead reduction effect

so of course its overhead is the highest. In MAM and MAMBO, due to Scribe's tree construction mechanism, more peers tend to subscribe to the root directly, than to other intermediary forwarders. Therefore, the root may also supervise more children than other parent peers. On the other hand, the POMP application requires the root to fulfil the accounting responsibility. In this case, MAM and MAMBO parent peers will report leaving/failures of their children to the root, which results in extra inbound communication overhead on the root. In short, the root in all the three approaches is a representative peer that can be used to measure a system's scalability.

Figure 5.11 demonstrates that as the online population increases from 50 to 1000, the workload of the service provider increases rapidly. However, for the roots of MAM and MAMBO, their workloads increase at a much slower speed, which means that MAM and MAMBO are less affected by the scale of the application, and thus they are more scalable than the conventional service-provider architecture. Furthermore, MAMBO is even more scalable than MAM, because its bushiness optimisation capability puts an absolute limitation on the number of children that the root may have.

Last but not least, it may appear strange that at the beginning of the experiment, the workload of the service provider is even lower than the root peer of MAM. However, this is because when the online population is small, e.g. 50 peers, the Scribe multicast tree

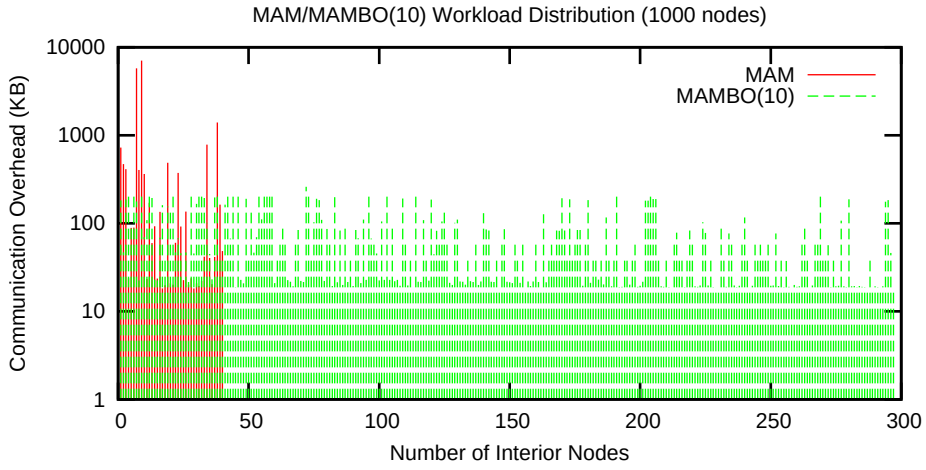

Figure 5.13: MAM & MAMBO(10) workload distributions

is shallow, and most of the peers subscribe to the root directly. In such a circumstance, MAM is not able to distribute the supervision work to multiple parent peers, and the root has to monitor all the application participants just like a service provider does. Moreover, MAM replaces explicit IAA messages with resubscribe requests, as discussed in Section 5.3. The size of a resubscribe request is usually bigger than a simple IAA message, because the former has to carry extra routing information. As a result, when the online population is small, the root for MAM could have higher inbound communication overhead than the service provider. However, this is never a problem for MAMBO, because the root for MAMBO will not accept more than 10 children, regardless of the actual online population.

5.5.4 Communication Overhead Reduction

The objective of this experiment is to carry out a quantitative study on the COR effect outlined in Section 5.4.2 under different degrees of peer talkativeness. The talkativeness parameter is defined as the probability of a peer to communicate with the root of a multicast tree in any maintenance interval. The experiment measures the inbound communication overhead imposed on the root peer with 500 application participants and talkativeness values ranging from 0% to 100%.

Figure 5.12 demonstrates that the communication overhead of the service-provider architecture reduces quickly as the peers' talkativeness increases, but this COR effect is not exhibited in MAM and MAMBO. When the mean talkativeness exceeds 73.5% per maintenance interval, the overhead of the service-provider architecture becomes lower than MAM, and when the mean talkativeness exceeds 94%, it even becomes lower than MAMBO.

The communication overhead of the MAMBO root is also lower than the MAM root because of the bushiness optimisation. For the same reason, the overhead of the MAMBO root is also more stable than the MAM root, since the amount of peers that directly subscribe to the MAM root may vary, whereas the MAMBO root consistently has 10 child peers when the online population is around 500 peers.

The experimental results validate the previous theoretical analysis about the COR effect. MAM is able to reduce communication overhead for applications with moderately talkative or quiet peers compared with the service-provider architecture, and MAMBO reduces the overhead still further.

5.5.5 Load-Balancing

This experiment investigates the load-balancing capability of the bushiness optimisation mechanism by comparing the shapes of a MAM and a MAMBO tree that are created for 1000 application participants, and by measuring the actual inbound communication overhead imposed on each parent peer.

Figure 5.13 shows that in MAM, only about 40 peers were selected to be parent peers, monitoring the other 960 peers in the system, whereas in MAMBO(Bushiness=10), about 300 peers were selected to be parent peers. It demonstrates that the shape of the multicast trees built by MAM and MAMBO are quite distinct, and the bushiness optimisation has a significant impact by ensuring that each interior peer only accepts up to a limited number of children.

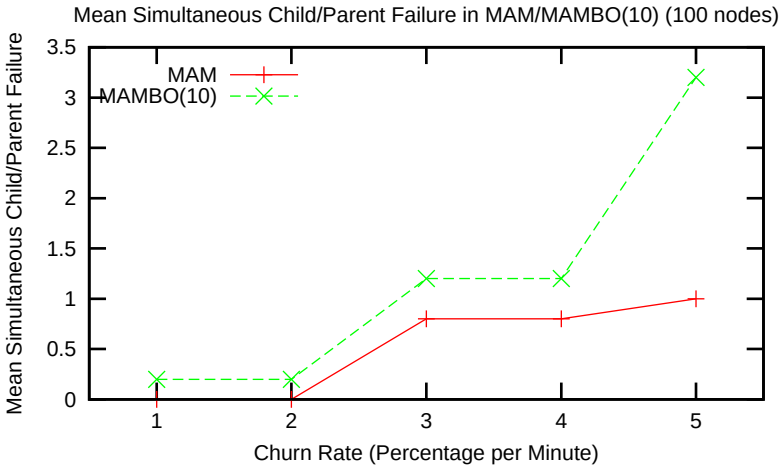

Figure 5.14: Mean number of exceptions in MAM & MAMBO(10)

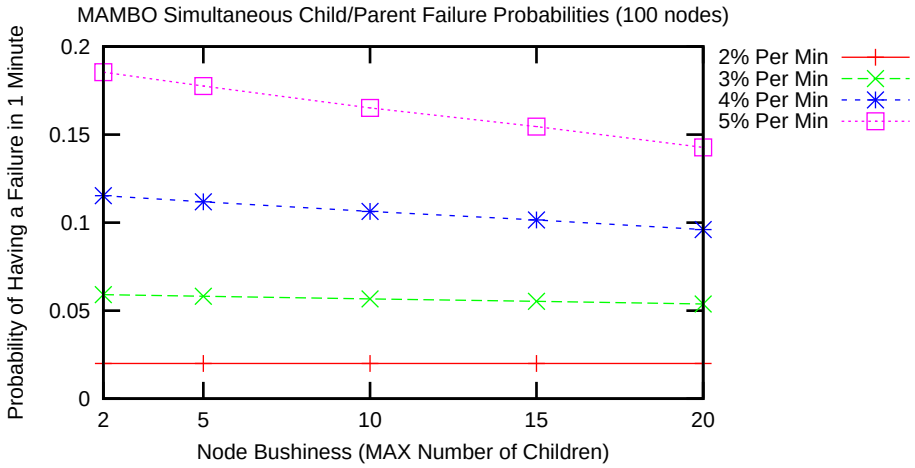

Figure 5.15: Simultaneous parent/child failure probabilities in MAMBO

Furthermore, while mapping supervision relationships onto a MAMBO tree, the total workload is fairly distributed among all the parent peers. However, in MAM the workload of some parent peers is about 100 times more than some other parent peers. It clearly indicates that the bushiness optimisation mechanism is quite effective at load-balancing.

5.5.6 Dependability Implication of Bushiness Optimisation

Though the bushiness optimisation provides a good improvement to load-balancing capability, its side effect on supervision is also observed in the experiments. Because MAM distributes the supervision responsibilities to multiple parent peers, sometimes it is unable to detect simultaneous child/parent failures as discussed in Section 5.5.2. Figure 5.14 further demonstrates that the bushiness optimisation mechanism can increase the probability for such exceptions to occur.

Figure 5.14 shows the mean number of supervision exceptions happened in MAM and MAMBO(Bushiness=10). Firstly, as an inherent design limitation, the exceptions happened to both MAM and MAMBO. Secondly, a higher churn rate implies that there are more peers leaving the system concurrently, so the probability for a supervision exception to happen is also higher. Last but not least, no matter what churn rate is applied to an experiment, the mean number of exceptions in MAMBO is always higher than in MAM. Figure 5.15 justifies these experimental results with a statistical analysis, where a higher degree of bushiness is helpful to reduce the exception probability.

MAMBO is an effective load-balancing mechanism, and it is helpful to further enhance the scalability of a P2P application. However, as a double-edged sword, it may also increase the probability of a supervision exception. Currently, several secondary policy plug-ins for MAMBO are being considered, which either attempt to exploit grandparents and siblings to enhance the parent-child supervision relationship, or to adopt a multi-tree strategy like Splitstream [37]. Before these secondary solutions become available, a program has been implemented, which suggests appropriate bushiness values according to the typical online population and the churn rate of a P2P application, as shown in Figure 5.15. Developers who want to apply MAMBO in their P2P applications may refer to the suggested bushiness value, and strike a balance between the workload that is to be imposed on each parent peer and the possible failure rate of the supervision mechanism.

5.6 Conclusion

Application-level multicast is an important research area investigating efficient communication infrastructures for sending data to a group of recipients. In the literature, tree-based application-level multicast systems have been implemented to facilitate P2P applications such as file sharing, multi-player online game, incentive mechanisms, and multimedia streaming. Some of these applications only provide a best-effort service, but others have higher membership management requirements.

This chapter proposes a new membership management mechanism, MAMBO, that reuses the ALM infrastructure in a P2P application to monitor and organise its membership, by overlapping the multicast tree with a supervision tree (Section 5.3). Unfortunately, a naive overlapping, MAM, may induce heavy workloads on some nodes (Section 5.5.5), so a tree Bushiness Optimisation is introduced to limit the maximum workload.

Both MAM and MAMBO have been implemented on FreePastry as policy plug-ins for Scribe (Section 5.4.1). With the demonstration application POMP we show the following advantages of MAMBO:

- MAM and MAMBO are more scalable than a conventional service-provider architecture, and MAMBO is more scalable than MAM (Section 5.5.3), e.g. the inbound communication overhead is $\frac{1}{30}$th of that in a service-provider architecture on a 1000-node P2P network. Hence MAMBO is more suitable for large scale P2P applications.
- MAM and MAMBO reduce communication overheads compared with a conventional service-provider architecture for applications with low levels of talkativeness. For example, on a 500-node P2P network, MAM outperforms the service-provider implementation up to 73.5% talkativeness, and MAMBO up to 94% talkativeness (Section 5.5.4).

- MAM and MAMBO keep the communication and computation overheads of providing membership management low. For example, the Scribe resubscription messages act as heartbeats indicating peer status (Section 5.4.1). MAMBO can further balance the supervision workload among application peers (Section 5.5.5).

- MAM and MAMBO keep the configuration overhead of providing membership management low by mapping the supervisor-worker relationship to the information forwarder-receiver relationship in the ALM infrastructure (Section 5.3). It is straightforward to deploy MAM and MAMBO as policy plug-ins in existing P2P applications that use tree-based application-level multicast (Section 5.4.3).

A limitation of MAM and MAMBO has been identified, i.e. its inability to detect the failure of a child if its supervising parent fails at the same time. Currently, a MAMBO user can partly compensate for this by choosing a bushiness configuration for a specific P2P application which minimises this risk (Section 5.5.6). In future work we aim to provide secondary policy plug-ins that remove this limitation.

The relationship between MAMBO and the Mediator framework discussed in Chapter 4 is that Mediator uses MAMBO as an efficient membership management mechanism for a ZM to monitor the presence and working states of its zone members, as discussed in Section 4.4.2. A ZM disseminates periodic heartbeat messages to its zone members via a tree-based application-level multicast. MAMBO reuses this communication infrastructure to distribute a ZM's supervision responsibilities to large numbers of information forwarders. As a result, the communication overhead that is imposed on a ZM can be reduced, and super-peer dependability of the Mediator framework can be enhanced without weakening its scalability.

— It's much better to do a little with certainty, and leave the rest for others that come after you.

Isaac Newton

Chapter 6

Deadline-Driven Auctions

6.1 Introduction

To adapt MMOGs to P2P architectures a range of design issues have to be addressed, as discussed in Chapter 3. NPC host allocation is one of these important issues. Typically a MMOG features considerable numbers of NPCs, i.e. virtual actors who drive continuing storylines or combat player characters. In conventional C/S architectures NPCs are hosted by centralised game servers, consuming significant processing power and network bandwidth. However in P2P architectures, such dedicated game servers are not available. As a result, a P2P MMOG needs to host its NPCs using resources available on game participant machines.

This chapter presents the design (Section 6.3), analysis (Section 6.4), extension (Section 6.5), implementation (Section 6.6) and evaluation (Section 6.7) of Deadline-Driven Auctions (DDA). DDA is a novel task mapping infrastructure for heterogeneous P2P environments, and is also an important component of the Mediator framework (Section 4.2.3). The relationship between DDA and the Mediator framework can be viewed from two perspectives. On the one hand, DDA completes the Mediator framework by providing a distributed NPC host allocation mechanism that is compatible with other sys-

tem components, such as interest-management and incentive mechanisms. On the other hand, the Mediator framework also facilitates the application of DDA by supplying a self-organising and dependable super-peer infrastructure.

Common NPC host allocation schemes are region-based [118, 91] or virtual distance based [29, 167, 86], as discussed in Section 3.4. The former hosts all the NPCs in a game zone using a single super-peer, and the latter allocates a NPC to the machine of a player, whose avatar is closest to the NPC. Compared with these approaches DDA offers the following advantages:

- Efficient Resource Utilisation - DDA shares NPC tasks among multiple application participants and guarantees that a NPC task is always allocated to a host with adequate resources. So, DDA is able to utilise available resources efficiently, and to bring the potential of a P2P system into full play.
- Real-time Resource Allocation - The fast pace of a MMOG usually requires a NPC to appear and start working within a few of seconds. Experimental results demonstrate that DDA is able to process large numbers of tasks within their deadlines, and to support a simulated P2P MMOG with the better part of 1000 players (Section 6.7.3).
- QoS for 1:N Interactions - DDA reduces communication latency among a NPC host and multiple ordinary players in the vicinity of that NPC, so as to provide better gaming experiences for 1:N interactions (Section 6.7.4).
- Cooperative Economic Model - DDA supports flexible resource selection policies, and it conveniently establishes a cooperative economic model that shares NPC tasks among competent resource providers fairly (Section 6.7.5). DDA's native incentive mechanism should persuade application participants to volunteer for NPC tasks, and thus maintains the viability of a P2P MMOG.

NPC Allocation Mechanisms	Category	Resource Utilisation	Task Allocation Overhead	Game Interactivity	Cooperative Incentive
P2P [118]	Region-based	Low	Low	Unoptimisable	None
Zoned [91]	Region-based	Low	Low	Unoptimisable	None
Colyseus [29]	Distance-based	Moderate	Unknown	Good for 1:1	None
AtoZ [167]	Distance-based	Moderate	High	Good for 1:1	None
Voronoi [86]	Distance-based	Moderate	High	Good for 1:1	None
DDA [66]	Task mapping	High	Moderate	Good for 1:N	DCRC

Table 6.1: Comparison of NPC host allocation mechanisms

6.2 DDA Related Work

Existing NPC host allocation schemes can be classified into static region based approaches and dynamic virtual distance based approaches. The former, e.g. [118] and [91], partition a game world into multiple regions, and assign each region a super-peer, which works as an authoritative server and hosts all the NPCs within the region. In contrast, dynamic virtual distance based approaches, e.g. [29], [167] and [86], distribute NPC objects to ordinary game participants, where the key idea is to allocate a NPC to the machine of the player, whose avatar is closest to the NPC in a virtual game world.

However, both of the approaches have several drawbacks in terms of the degree of utilisation of resources, overhead for task allocation, game interactivity optimisation and support for cooperative incentive. Table 6.1 systematically compares DDA to five pieces of representative related work from these perspectives.

Firstly, region based approaches cannot make use of application participants' resources efficiently, because they only employ one super-peer to host the NPC objects of a whole region. As a result, excessive computation and communication workloads are put on a single super-peer, whereas resources available at other peers in that region are not exploited. Dynamic virtual distance based approaches are relatively better at sharing NPC

tasks among multiple application participants. However, a potential problem in such approaches is that the player that is closest to a NPC may not have adequate resources to host the NPC for other players. In contrast, DDA's task mapping mechanism takes players' actual resource availability into consideration, and guarantees that NPC tasks are always allocated to capable resource providers. Therefore, DDA can utilise available resources in a P2P system more efficiently than the related work.

Secondly, the overhead for achieving a NPC host allocation mechanism should not be neglected. In region based approaches, all NPC tasks are assigned to the same super-peer, so no decision making is required, and only a little overhead is incurred by super-peer selection process. Comparatively, the computation of accurate NPC host allocation is more expensive in virtual distance based approaches. For example, AtoZ [167] uses Mahalanobis distance in the domain of quadratic discriminant analysis [21], and Voronoi [86] requires each peer to construct and maintain a Voronoi diagram by itself. Even worse, because a large proportion of the players in a MMOG are constantly moving, switches of NPC host may be frequent, and thus communication overhead can also be high. In comparison with the analysis of virtual distance, DDA's distributed matchmaking process is not computational intensive, and once a NPC is allocated to a game participant, the hosting relationship remains stable, unless the NPC is destroyed, or the host needs to leave the system. The experimental results in Section 6.7 demonstrate that DDA is able to process large numbers of NPC tasks within their deadlines, and to support a simulated P2P MMOG with approximately 1000 players on a single computer. However, Colyseus [29] and AtoZ [167] have only been evaluated with a few players and game objects, so their practicability and performance at a large scale are unknown. On the other hand, Voronoi [86] has to resort to dedicated server nodes when the overhead for analysing NPC hosting relationships becomes overly high.

Thirdly, game interactivity is another important criterion for comparison. A NPC host should have low communication latencies with other players that interact with the NPC, so as to guarantee a smooth gaming experience for a P2P MMOG. From this point of view, region based approaches are unoptimisable, because in each region a single super-

peer is selected to host all the NPCs, regardless of which players will interact with them. Virtual distance based approaches are inherently optimal for 1 to 1 interactions. When the player that is closest to a NPC happens to be the only player that interacts with the NPC, the player can host the NPC by itself and avoid communication latency completely. However, most NPCs in a MMOG can have a real-time effect on players on several hosts at a time. In this case, all the players need to communicate with the NPC host, and virtual distance based approaches cannot ensure that the latency for each player is equally low. Contrarily, DDA aims at reducing communication latency among a NPC host and multiple ordinary players in the vicinity of that NPC, so as to provide better gaming experiences for 1 to N interactions (Section 6.7.4). Though DDA requires a third party NPC host even in 1 to 1 interactions, the communication latency between a player and a NPC host is kept low. Furthermore, the use of a third party NPC host should also help to reduce the opportunities for unscrupulous players to abuse their hosting of NPC objects to their own advantage.

Last but not least, when a P2P MMOG relies on game participants to carry out various functional and administrative tasks, an incentive mechanism becomes crucial to persuade application participants to contribute their resources, and to maintain the viability of the system (Section 3.7). Unfortunately, none of the related work has discussed their supports for incentive mechanisms. So, another advantage DDA provides is that it supports flexible resource selection policies, and it conveniently establishes a cooperative economic model that shares NPC tasks among competent resource providers fairly (Section 6.7.5). Such an economic model enables a DCRC-like [77] accounting mechanism to identify and discourage free-riders in a P2P MMOG. These free-riders are selfish players who attempt to consume more resources than they contribute as well as shoulder fewer responsibilities than other players.

In conclusion, DDA is a dynamic task mapping mechanism for heterogeneous P2P environments that is different from existing region based or virtual distance based NPC host allocation approaches. Compared to related work, DDA is designed to utilise game participants' resources well and always distribute NPC tasks to capable resource providers.

It is designed to allocate computing resources efficiently for large numbers of real-time NPC tasks and to support gaming interactivity by keeping the communication latency among NPC hosts and ordinary players low. Finally, it supports flexible matchmaking policies, and with a friendly incentive policy, can establish a cooperative economic model that helps motivate participants to contribute their resources to the system. The following sections in this chapter present the design of DDA, a theoretical analysis of its model, as well as the implementation and evaluation of its prototype.

6.3 Design

6.3.1 System Model

DDA is a novel task mapping infrastructure for heterogeneous P2P environments. Traditionally Heterogeneous Computing (HC) refers to the coordinated use of various resources with different capabilities to satisfy the requirements of varying task mixtures [70, 32]. A resource management system is employed to maximise the cost effectiveness of an HC system by appropriately assigning resources to tasks (i.e. matching) and ordering the execution of tasks on the resources (i.e. scheduling) [102]. This procedure of matching and scheduling is defined as a "mapping" [102]. The design of DDA's mapping infrastructure is different from a traditional HC system because of the following reasons:

Characteristics of tasks & resources In traditional HC systems, an application is assumed to be composed of one or more independent tasks, and some of the tasks may be further decomposed into multiple communicating subtasks [70]. The subtasks have data dependencies among them, but are able to be assigned to different machines for execution. The resources, on the other hand, are dedicated to an HC system, such as a computer cluster that is owned by a single institution. In this case, a resource manage-

ment system is given full control of these resources, and able to schedule the execution of a task on any computer at any time.

In contrast, DDA is customised for open, dynamic and fully distributed P2P applications. It is assumed that in such applications a large proportion of participants have spare computing resources besides running their own tasks, and that the participants are willing to contribute these resources by running public tasks for the collective welfare. A public task is usually independent, such as to provide a service for other participant peers during a specific period of time. However, because a participant may join and leave a P2P application freely, and its resource availability may change over time, DDA does not have the knowledge about what type and amount of resources are available on which participants. As a result, DDA separates the matching and the scheduling, where application participants carry out the scheduling by themselves (e.g. Mediator local scheduling, Section 6.3.3) and report temporary resource availability information to resource management super-peers, which carry out the matching accordingly (e.g. Mediator zone-level matching, Section 6.3.4).

Goals & methods of mapping A traditional HC system is considered to be oversubscribed, when there are always tasks waiting to be mapped, and it is unlikely that all the tasks will be completed within their deadlines. So, the priority of each task is evaluated, and the goal for the mapping is to maximise the cost effectiveness of an HC system by meeting the deadlines of larger numbers of important tasks within a certain time interval. However, the problem of producing an optimal mapping has been shown to be NP-complete [90], and thus various heuristic techniques are proposed as methods for finding near-optimal mappings [102].

The goal for DDA is quite different. Above all, it is assumed that the spare computing resources available on the participants of a P2P application are always adequate to satisfy the requirements of public tasks. If this were not so, the P2P application type would not be viable. Furthermore, the priority of a task is determined by the deadline for the task to be mapped, whereas other metrics of importance such as the deadline

for the task to be completed are seldom considered. Besides quantitative requirements such as the amount of resources required, a task may also specify a set of qualitative requirements on potential resource providers' trustworthiness, dependability, quality of service and so on. In this case, the suitability for multiple resource providers, which all have adequate resources to run a task, can still be compared and ranked. So, the goal for DDA's mapping is to assign all the pending tasks in a P2P application to satisfactory resource providers within their deadlines. The method of achieving this goal is through opportunistic ***matchmakings*** and ***auctions***.

A ***matchmaking*** refers to the process of discovering a satisfactory resource provider, which both satisfies a task's quantitative requirements and ranks high according to the task's additional qualitative requirements. DDA's matchmaking is opportunistic, because the mapping of a task has to be completed within the task's deadline. Consequently, a relatively more competent resource provider that is discovered in due course at matchmaking time is regarded as satisfactory, though it might not be an ideal choice if more time was available or the mapping was required at a different time. In large scale P2P applications DDA may distribute matchmaking responsibilities to multiple matchmaker super-peers, each managing a subset of the resource providers. Given a task request, each matchmaker proposes a locally optimised resource provider, and an auction is used to determine the most satisfactory resource provider among these offers.

An ***auction*** in economic theory refers to any mechanism or set of trading rules for exchange. A typical format of auction comprises offering an item up for bid, taking bids, and then selling the item to the winning bidder. Auctions have been widely used in market-based P2P resource sharing systems, e.g. Popcorn [137], Spawn [158] and Tycoon [106]. These systems support their users to trade computing resources for virtual currency, and users bid for resources with the rate that they are willing to pay for the resource. In contrast, DDA supports multiple matchmakers to bid for tasks with locally optimised resource providers. A resource provider that is evaluated to be more suitable for running a task is kept as a winning bid, whereas other less competitive bids are rejected immediately. An auction's time limit is determined by the deadline for the task

that is up for bid. When there are no more matchmakers to bid for the task, or the deadline for the task is sufficiently near, the auction is closed and the task is assigned to the winning resource provider.

In summary, the system model for DDA involves three parties:

1. Work Source The work source is the origin of tasks that need to be run by the participants of a P2P application. The work source is also an auctioneer that distributes task requests to multiple matchmakers in the system, accepts bids and determines the final winning bid. DDA supports general P2P applications with real-time computational or interactive tasks. A computational task may specify the computing resources that it requires, such as CPU processing power, memory and storage capacity. An interactive task may specify the communication resources that it requires, such as network bandwidth and the maximal communication latency. Furthermore, both kinds of tasks may specify a deadline, before which an appropriate resource provider must be located, as well as additional qualitative requirements on a resource provider's trustworthiness, dependability and quality of service.

2. Resource Providers The resource providers are application participants that have spare resource on their machines and would like to contribute the resource to the application by running various tasks. Typically, resource providers in a P2P system have two important characteristics - heterogeneity and non-dedicated resources [164]. The former means that the resource providers may have different hardware configurations and network connections. As a result, their capabilities of processing tasks may differ significantly in terms of computation and communication. Non-dedicated resources, on the other hand, indicates that a resource provider may share its resource among multiple applications concurrently, and thus its performance may vary over time. In this case, DDA allocates tasks according to each resource provider's individual capability, as well as the latest resource availability.

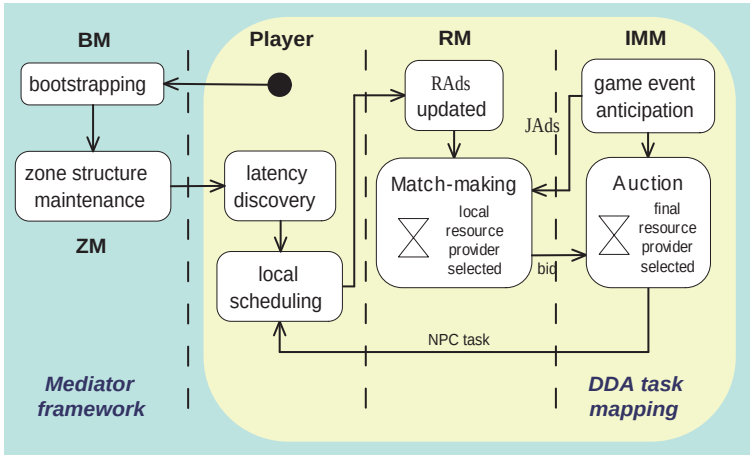

Figure 6.1: The Mediator framework & Deadline-Driven Auctions

3. Matchmakers The key idea in DDA is to bridge between resource requirements and their availability using opportunistic and distributed matchmaking via resource matchmakers. Firstly, the work source disseminates resource requirement information to a set of matchmakers. Secondly, each resource provider requests tasks by uploading its resource availability information to a nearby matchmaker. Thirdly, a matchmaker matches resource requirements to resource offers, and proposes a locally optimised resource provider for each task to the work source. Finally, when the work source has received proposals from all the matchmakers, or the deadline for a task is sufficiently near, it closes the auction and determines which resource provider is finally selected to run the task.

6.3.2 Mediator DDA

DDA is primarily designed to support NPC host allocation in P2P MMOGs, and specifically in the Mediator framework (Chapter [64]). Figure 6.1 depicts the roles of the three parties when DDA is applied to the Mediator framework, and the components of DDA are fulfilled as follows:

Work Source The Mediator work source is the virtual game world that constantly generates requests for NPC objects. Conceptually, a NPC object is an indivisible, computational and interactive task, because:

- It can only be efficiently hosted by a single computer.
- It consumes processing power, as a NPC is associated with an AI program that determines the NPC's actions. For example, a monster NPC may determine its move by examining the surrounding terrain and the positions of nearby players.
- It needs to interact with players. For example, a monster NPC may engage in combat with a set of players, and exchange real-time gaming events with them.

The IMM for each game zone is in fact an oracle for the game world, as it anticipates game events and issues NPC task requests according to game scenarios. An important responsibility of an IMM is to allocate each NPC task within its deadline to a suitable resource provider that has adequate computing resources, and is able to provide low communication latency to player characters in the vicinity of the NPC.

Resource Providers The resource providers in the Mediator framework are ordinary game participants. Driven by an incentive mechanism, the game participants carry out a series of local scheduling activities (Section 6.3.3), and may decide to volunteer for hosting NPC tasks, serving various super-peer roles, or registering as super-peer backups.

Matchmakers The RMs for each game zone serve as DDA resource matchmakers. On the one hand, they buffer NPC task requests from the IMM, and on the other hand, they receive resource offers from ordinary zone members. Each RM matches a NPC task request to all existing offers, and then proposes a locally optimised resource provider to the IMM, which in turn compares the candidates and determines the final NPC host.

6.3.3 Local Scheduling

Local scheduling refers to the activities carried out by a peer to manage its local computing resources, and to contribute spare resource to a P2P application. Typical local scheduling activities in Mediator DDA include:

Self-evaluation of Computing Capability A peer may evaluate its own resource availability, and estimate what kind and amount of tasks it can afford. For example, a ZM in the Mediator framework issues super-peer requirements to its zone members periodically (Section 4.3.2), specifying the processing power and network bandwidth required by different super-peer roles. By receiving such advertisements, each ordinary zone member carries out self-evaluation to tell whether it is capable of a super-peer role. This self-evaluation process is crucial for dealing with the resource heterogeneity in a P2P system.

Volunteering for Super-Peer Work A peer may decide whether to volunteer for super-peer work, if it qualifies for a super-peer role. For example, a resource-rich peer that qualifies for a ZM role can register at the current ZM to become a ZM backup. Where the current ZM is leaving the zone, or is about to be overloaded, it will give up its role to a suitable backup peer, or divide the game zone into multiple sub-zones and appoint new ZMs for each of them from backup peers. Similarly, qualified peers can also register as backups for the IMM or the RMs, in case some of them leave or fail.

QoS Optimisation A peer may need to optimise its quality of service according to application-specific requirements. For example, real-time interactions in a P2P MMOG are susceptible to high communication latency. So, in order to provide smooth gaming experiences, Mediator DDA requires resource providers to find out their communication latencies with other players in the same game zone, which becomes an important part of a peer's local scheduling. Concretely, the ZM for each game zone publishes heartbeat

messages periodically for zone structure maintenance purposes. Such messages carry a list of existing zone members, so that a peer can check whether new zone members have shown up. If so, the peer sends out a set of `Ping` messages to the new zone members, and they reply with `Pong` messages, so that communication latencies among them can be measured.

Requesting Tasks A peer may request new tasks by uploading its resource availability information to a matchmaker. For example, in the Mediator framework an ordinary player may request NPC tasks by uploading its resource availability information to a nearby RM. Due to the resource non-dedication property, a resource provider should update its availability information at the matchmaker periodically, so that the matchmaker always works with information that is reasonably up-to-date.

Condor-like Matchmaking Mechanism

DDA resorts to a Condor-like [113] matchmaking mechanism for resource discovery, and represents both the available resources and the task requests using ClassAds. A ClassAd is a set of uniquely named expressions, which can be used to describe the characteristics and constraints of machines and jobs [133]. In Mediator DDA, a ClassAd that describes a usable resource is called an RAd, and a ClassAd that describes a job is called a JAd. Figure 6.2 shows the structures of these two kinds of ClassAds.

RAd Structure: The **Owner** field contains a resource provider's PeerId as a 128bit code. The **PRI** and **Rep** fields respectively indicate the resource provider's current virtual credits and reputation score. The **CPU**, **RAM**, and **BW** fields reflect a peer's available processing power, memory and network bandwidth. A resource provider is able to specify a minimum task weight that it accepts in the **Mini** field, because on the one hand a powerful provider may not want to be bothered by tiny tasks, and on the other hand, less competent providers would be starved of contribution opportunities if all tasks in

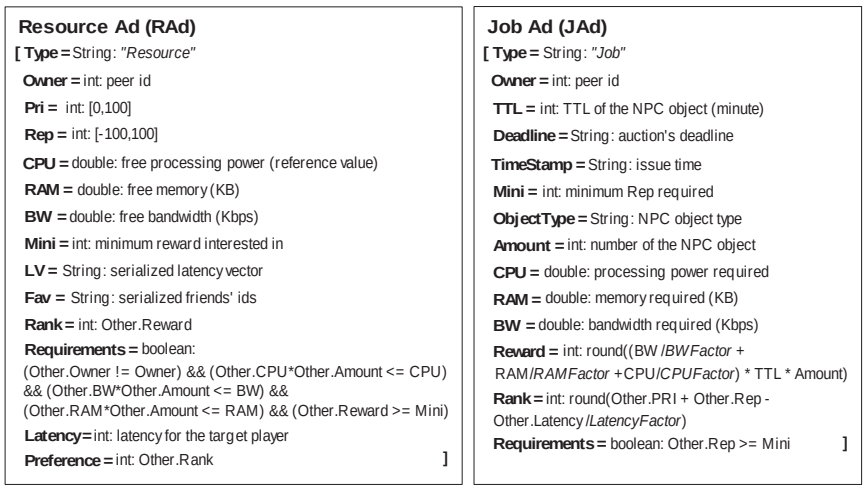

Figure 6.2: Structures of ResourceAd and JobAd

their power were seized by stronger peers. The **LV** field is a serialized HashMap with PeerIds as keywords, and communication latency values as contents. The **Rank** and **Requirements** fields are required by Condor's matchmaking mechanism. By the evaluation of those two fields, it is determined whether a given resource provider is able to host a specific task or not, and to what extent both parties are a good match for each other. The last two fields, **Latency** and **Preference** are completed by the RM that carried out the matchmaking. A RM finishes the RAd from the locally optimised resource provider, and passes it to the IMM as a bid for the corresponding NPC task request.

JAd Structure: The **Owner** field reflects a set of PeerIds, for whom RMs should minimise communication latency while selecting the locally optimised resource provider. The owners of a specific NPC task request are determined by the IMM that issued the JAd, according to a game event anticipation algorithm. If a NPC task was triggered by a single player for a 1:1 interaction, that player would be identified as the owner of the task request. Otherwise, if a NPC was spawned for a 1:N interaction, all of the players that are in the vicinity of the NPC would be identified as the owners of the task request. The **TTL**, **Amount**, and **ObjectType** fields respectively represent the life time, the cardinality, and the type of a NPC. These three fields will be used for a selected resource

provider to initialise the NPCs. Furthermore, the IMM can specify a **Mini** field in a JAd to restrict the minimum reputation score for a required resource provider, according to the significance of the NPC to be hosted. The **Deadline** and **TimeStamp** fields are utilised by DDA to make sure that a resource provider is located in due course. Finally, **CPU**, **RAM** and **BW** fields describe the weight of a NPC task.

DCRC-like Incentive Mechanism

Local scheduling is driven by a DCRC-like incentive mechanism [77], which on the one hand charges peers in term of credits according to the time that they play the game, and on the other hand, rewards peers according to the amount of resources that they contribute to the system. Ideally, a peer that actively contributes usable resources to the system should be able to earn enough credits to pay for its playing time. In contrast, free-riders will become poorer and poorer, and finally be identified and discouraged.

However, a potential problem with this incentive mechanism is that if resource-rich peers eagerly volunteer for tasks, less competitive peers will be starved of opportunities for earning credits, and as a result, they could come to be regarded as free-riders unfairly. To cope with this problem, DDA adopts a friendly matchmaking policy (Section 6.5.2) that fairly shares credit earning opportunities among all competent resource providers.

Cheating Mitigation

Currently, cheating mitigation is not a primary focus of DDA's design, but obviously it is possible for unscrupulous peers to cheat on local scheduling. For example, in order to earn more credits, a peer may lie about the spare computing resource it can provide, or upload its resource availability to multiple RMs. As a result, the peer will be allocated tasks that are beyond its capability, which consequently damage other peers' gaming experiences. One avenue of future work is to enhance DDA's security by discouraging disadvantageous peer behaviour using a distributed reputation system, which enables a

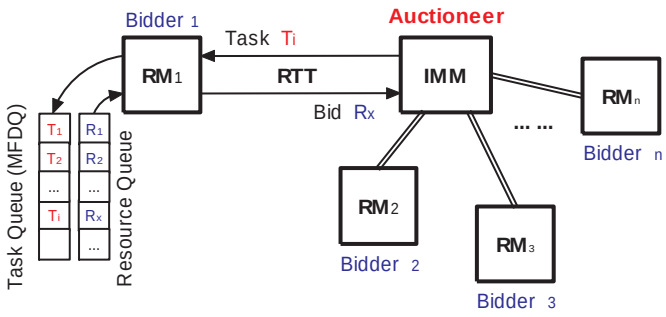

Figure 6.3: DDA zone-level scheduling

RM to favour more dependable and trustworthy resource providers during a matchmaking process.

6.3.4 *Zone-Level Matching*

Zone-level matching refers to the collaboration among the work source (i.e. IMM) and multiple resource matchmakers (i.e. RMs), as depicted by Figure 6.3.

Each RM maintains two queues: a resource queue (RQ) for storing resource availability information (i.e. RAds) from resource providers, and a task request queue (TQ) for buffering NPC task requests (i.e. JAds) given by the IMM.

1. RQ An ordinary player that is willing to request NPC tasks will firstly complete its local scheduling activities, and then upload an RAd to a nearby RM to report its resource availability and communication latencies with other players. The RM stores the RAds that it has received in the RQ, and for each pending NPC task request, the RM scans the RQ to locate a locally optimised resource provider that has adequate resources, as well as the lowest mean communication latency with the target players specified in the task request. Finally, the RM returns the selected RAd to the IMM as a bid for the corresponding NPC task request. In this matchmaking process, the following aspects need to be emphasised:

- The Length of RQ - Every time an RM receives an RAd, it can recognise whether the RAd is from a familiar resource provider, or a stranger. In the former case, the RAd is regarded as an update, and will replace that provider's existing RAd in the RQ. In the latter case, the RAd is regarded as a new subscription to the RM, which will result in the length of the RQ being increased by 1. However, an RQ cannot be arbitrarily long, because lengthening it will both prolong the time that the RM takes to process a NPC task request, and impose more workloads on the RM. Therefore, the Mediator framework allows application developers to set a limit on the maximum length of an RQ. Once an RM is fully subscribed, the ZM will remove it from heartbeat messages, and may promote more RMs from super-peer backups for load-balancing purposes (Section 4.3.2).

- Matchmaking Time - Suppose that the maximum length of an RQ is l RAds, and it takes t milliseconds for an RM to match each RAd to a NPC task request, it will take $l * t$ milliseconds in total for the RM to complete matching all the RAds to decide which one is the most appropriate. So, $l * t$ is referred to as the maximum matchmaking time, which plays an important role in the design analysis (Section 6.4). Some preordering of RAds might reduce the search time a little, but would incur the overhead of computing a preordering for every new RAd received.

- Matchmaking Policy - Previously, it has been mentioned that for each NPC task request an RM will by default select a resource provider that has adequate computing resources and the lowest mean communication latency with the target players. This selection criterion always puts forward a potential NPC host that is most likely to provide adequate game interactivities, whereas experimental results demonstrate that it also has a negative side effect on the incentive mechanism (Section 6.7.5). To deal with this problem, DDA supports flexible matchmaking policies for RMs to overload the definition of a "locally optimised resource provider". More details about matchmaking policy will be discussed in Section 6.5.2.

2. TQ A RM buffers NPC task requests (i.e. JAds) issued by the IMM in the TQ. Because each task request comes with a deadline, before which the task must be mapped, a TQ suits being made a priority queue, where task requests are ordered on an earliest deadline first (EDF). EDF is a scheduling algorithm that selects the task closest to its deadline as the one to be processed next. The processing cycle of the RM involves retrieving the JAd with the shortest deadline from the TQ, scanning the RQ to find the most adequate resource provider, and returning it to the IMM as a bid. From the IMM's perspective, for each NPC task request that has been sent out, it opens an auction that waits for resource offers to be returned by RMs. When the IMM has received offers from all the RMs, or the deadline for the NPC task is sufficiently near, the IMM closes the auction and determines which resource provider is finally selected to host the NPC.

Here, a potential problem is that there are likely to be multiple offers, only one of which can win the auction, and the IMM will notify the rest of the RMs that their bids have been rejected. Before the RMs receive this notification, it is safer for them to proceed to the next matchmaking cycle without considering the proposed resource providers, because if a resource provider won the auction, its resource availability would change. Consequently, many proposed resource providers that did not win the current auction may also lose the opportunity to be considered for the next few auctions. This is not only unfair to the resource providers, but may also damage the overall resource quality attained by the system. To deal with this problem, a multilevel feedback delay queue (MFDQ) is proposed as the infrastructure of a TQ, instead of an intuitive EDF queue. More details about the MFDQ will be discussed in Section 6.5.1.

6.4 Design Analysis

The purpose of this section is to develop a simple mathematical model of DDA and to analyse whether DDA is efficient enough to keep up with the fast pace of a MMOG by meeting the deadlines for large numbers of NPC task requests.

6.4.1 Deadline for Player-Triggered Tasks

The deadline for a player-triggered task T is a specific time point D, before which an appropriate resource provider must be located through zone-level matching. For example, a player is approaching a shop and will arrive in 5 seconds. Accordingly, a merchant NPC is triggered, which must show up in the shop and start working before the player's arrival. This requirement is captured in Equation 6.1, in which $C(T)$ means the completion time for task T.

$$C(T) < D \tag{6.1}$$

According to Figure 6.3, $C(T)$ comprises two periods of time: the round-trip time (RTT) between the IMM and an RM, and the time that an RM takes to discover a locally optimised resource provider. To simplify the analysis, it is assumed that by carefully selecting super-peers from candidates, each RM has an equally short RTT with the IMM. Furthermore, the RQs maintained by different RMs are equally long, each containing l RAds. If it takes t milliseconds for an RM to match one RAd to the JAd, it will take $l * t$ milliseconds in total for the RM to complete matching all the RAds to decide which one is the best. So, for task T:

$$C(T) = RTT + l * t < D \tag{6.2}$$

Equation 6.2 expresses the time restriction on a single NPC task request, whereas typically there are multiple task requests waiting to be processed in a TQ. Suppose that T_i is the i th task buffered in a TQ, since the previous $i - 1$ tasks all have shorter deadlines. As a result, it takes $(i - 1) * l * t$ milliseconds before the RM actually starts to process T_i. So, if T_i's deadline is D_i, its completion time should satisfy:

$$\forall_{i \in 1 \ldots n_\bullet} C(T_i) = RTT + i * l * t < D_i \tag{6.3}$$

6.4.2 Deadline for Periodically Respawned Tasks

Besides player-triggered tasks, there are also considerable numbers of NPCs that are periodically respawned in a MMOG, such as monsters that combat player characters (PCs). The spawning and respawning of a NPC object has been discussed in Section 2.2.2. Currently, well-known commercial MMOGs [7, 10] employ a timer-based approach that respawns NPCs to keep the number of NPCs and PCs to a stable ratio. In other words, if the number of players in a game zone is P, the NPC to PC ratio is R, and on average a NPC's life time is TTL seconds, $\frac{P*R}{TTL}$ NPCs should be respawned every second. Furthermore, if a timer is set to respawn NPCs every Int th second (i.e. the respawning interval), each time $\frac{P*R*Int}{TTL}$ NPCs are respawned.

In fact, the respawning interval determines the deadline for a set of NPC tasks, because all these NPCs must appear in the game world and start working within that interval, i.e. for all the NPC task requests that are respawned in the same interval, $D_1 = D_2 = ... = D_i = Int$. When such a set of task requests are buffered in a TQ, if the last request can be processed before the deadline, previous tasks can meet this deadline as well. So, Equation 6.3 is changed to:

$$C(T_{last}) = RTT + \frac{P * R * Int}{TTL} * l * t < Int \tag{6.4}$$

6.4.3 Combining Both Kinds of Tasks in a Respawning Interval

In this section, a respawning interval is used as a discrete time unit for further discussions. In other words, we can imagine that snapshots of a TQ's contents are taken for the duration of every respawning interval. It is highly likely that both player-triggered and periodically respawned task requests may coexist in the same snapshot. Because the deadline for respawned task requests is fixed, which is the end of the current respawning interval, deadlines for player-triggered task requests that fall into the same interval

Variable	Meaning	Nominal Value
P	zone population	
R	NPC:PC ratio	5:1
TTL	NPC life time expectation	300 (second)
r	event triggering rate	1/60 (per second)
Int	respawning interval	
RTT	round trip time	0.5 (second)
l	RM resource queue length	50 (RAds)
t	matchmaking time	1 (ms per RAd)

Table 6.2: DDA model variables & nominal values

are likely to be earlier than the former. Therefore, player-triggered task requests usually have higher priorities, are placed in the front of the TQ, and will be processed by the RM first. In this case, whether these requests can meet their deadlines is tightly related to the game event anticipation algorithm, which is beyond the scope of this thesis. In contrast, periodically respawned task requests have lower priorities, and their processing can be delayed. Hence, the analysis mainly focuses on whether the last NPC task request in each respawning interval can be processed in due course.

Assuming that each player triggers r NPC tasks in every second, the number of players in a game zone is P, and the length of a respawning interval is Int, there would be $r * P * Int$ player-triggered tasks in total. Since these tasks are processed first, the completion time of the last periodically respawned NPC task in Equation 6.4 is changed to:

$$C(T_{last}) = RTT + (\frac{P * R}{TTL} + r * P) * Int * l * t < Int \tag{6.5}$$

Table 6.2 lists all the key parameters of the DDA model, together with nominal values typical of commercial MMOGs [7, 10]. Applying the nominal values to Equation 6.5 creates Equation 6.6.

$$(600 - P) * Int > 300 \tag{6.6}$$

6.4.4 DDA Model Summary

We make the following key observations from our model:

- Based on the assumptions in Table 6.2, DDA can support up to 600 peers in each game zone, which corresponds to a zone size of around 500 peers in many typical MMOGs (Equation 6.6).
- When implementing a P2P MMOG with a zone size of 500 peers, the minimum respawning interval is 3 seconds (Equation 6.6).
- On average a resource provider may obtain a NPC task in every 10 respawning intervals, so the credits rewarded for running a task should cover the payment charged for corresponding playing time.

6.5 DDA Extensions

6.5.1 Reducing Resource Tie-up with a Multilevel Feedback Delay Queue

Figure 6.4 demonstrates a potential "resource tie-up" problem in DDA's design. Suppose that there are 200 peers in a zone, and a respawning interval of 6 seconds is used. Accordingly, at the beginning of every respawning interval, the IMM sends out 20 tasks (i.e. T_1 to T_{20}) to corresponding RMs, specifying the deadline as 6 seconds. For T_1, RM Alice proposes RAd R_x to the IMM as a bid, then carries on processing T_2. Unfortunately, at the IMM end, R_y proposed by RM Bob is better than R_x, so R_x is rejected and returned to Alice. In spite of the time for IMM's decision making, it will take at least

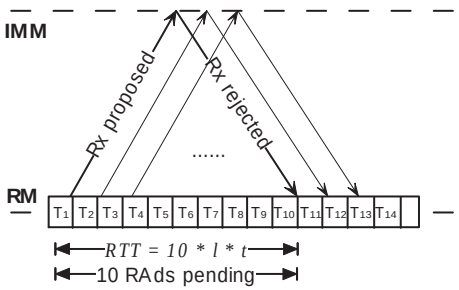

Figure 6.4: The resource tie-up problem.

RTT time before Alice finds out that her bid has been rejected. In Expression 6.2, $l * t$ is too small compared to RTT. Consequently, before R_x is actually returned to Alice, 10 matchmaking jobs have been completed without taking R_x into consideration. This is unfair to the resource provider and may also damage the overall resource quality attained by the system, as R_x could be a good match for some NPC task that it has missed. Even worse, as shown by Figure 6.4, it is possible for up to 10 RAds to be tied up throughout a respawning interval. Considering that a RM only has about 50 RAds in its RQ, this leads to a 20% tie-up rate.

To address this problem, a RM should anticipate the rejection or acceptance of its bids, and accordingly update the contents of related RAds, rather than simply removing them from the RQ. If a RM always presumes that its bids will be rejected by the IMM, it is referred to as a "pessimistic strategy", and the RM will carry on to the next matchmaking job without updating the RAds that it has proposed. Contrarily, a RM can adopt an "optimistic strategy" by always presuming that its bids will be accepted by the IMM. In this case, every time the RM makes a proposal to the IMM, it subtracts the resources to be consumed by the corresponding task from the original RAd, puts the new RAd back to its RQ, and then moves on to the next matchmaking job.

Comparatively, the optimistic strategy is safer than the pessimistic strategy, because in the latter there is a risk for the same resource provider to be proposed for a number of tasks and win multiple of them simultaneously. Though the resource provider is capable of hosting any individual task, it may not have adequate resources to host all of them.

As a result, it would be possible for some resource providers to be overloaded under the pessimistic strategy, whereas this problem is avoided with the optimistic strategy.

The optimistic strategy is not an ultimate solution to the resource tie-up problem, as it is only effective at improving the matchmaking efficiency for resource-rich peers that have excessive resources even after being proposed several times. In contrast, once a resource restricted peer is proposed to the IMM as a bid and a set of resources is subtracted in advance, the peer may no longer have adequate resources to be considered in the following matchmaking process. In this circumstance, the peer has to be tied up, until the RM receives the IMM's notification about whether the bid is finally accepted.

In fact, though theoretically DDA can support up to 600 peers in each game zone, the actual population might be smaller, especially when dynamic zoning is supported. In the previous example, when the average zone population is 200 and the respawning interval is 6 seconds, a RM can finish its matchmaking process for the 20 respawning tasks within $0.001 * 50 * 20 = 1$ second, which is long before the deadline of 6 seconds. Hence, it may be advantageous for a RM to introduce delays in between its matchmaking. For example, if a 150 ms delay is added at the end of each matchmaking, it will take a RM $0.001 * 50 + 150 = 200$ ms to complete a task, and 20 tasks will take 4 seconds, which is still ahead of the deadline of 6 seconds. With this amount of delay, a resource provider that is tied up during the matchmaking will only miss the next 4 tasks, instead of 10. In other words, there are at most 4 RAds being tied up throughout every respawning interval, which is a 60% improvement in terms of resource tie-up rate than before.

So, DDA should employ an optimistic matchmaking strategy while proposing a resource-rich peer to the IMM as a locally optimised resource provider. On the other hand, DDA should also require a RM to engage in delays in between its matchmaking. Considering that in every respawning interval a RM has to handle large numbers of periodical NPC tasks as well as player-triggered tasks with various deadlines, the scheduling algorithm for a RM could be complex. Hence, a useful extension to DDA would be a multilevel feedback delay queue (MFDQ), which is proposed to facilitate a RM to schedule its

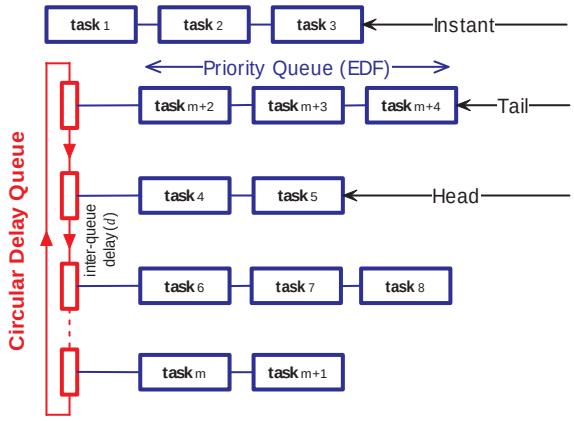

Figure 6.5: Structure of a Multilevel Feedback Delay Queue

matchmaking more effectively. A MFDQ can be used as the data structure of a RM's TQ for buffering NPC task requests, instead of an intuitive earliest deadline first (EDF) queue. Experimental results demonstrate that a MFDQ can alleviate much of the resource tie-up problem and increase the probability for DDA to locate a more appropriate NPC host (Section 6.7.4).

At the first glance a MFDQ is similar to a multilevel feedback queue for Unix process scheduling, where the main difference is that in the MFDQ, each run queue is executed only when its delay has expired. As shown by Figure 6.5, an individual run queue is a priority queue that implements EDF. If the inter run queue delay is d, then a MFDQ comprises $n = ceil(Int/d)$ run queues in total. Each run queue represents a different urgency level. Queue `Instant` buffers the most urgent tasks, which are processed instantly. Other run queues are arranged in a circle, with a `Head` pointer indicating the queue to be executed next. When a inter-queue delay has expired, tasks in queue `Head` are moved to queue `Instant`, and the `Head` pointer jumps to the next run queue.

A MFDQ mainly supports two operations:

1. $add(Task\ t)$ This enqueue operation inserts a new task into the MFDQ using the following algorithm:

```
add(Task t){
  t.enqueue_deadline = t.original_deadline-RTT-l*t;
  t.urgency_level = floor(t.enqueue_deadline/d);
  if(t.urgency_level == 0)
    Instant.add(t);
  else if(t.urgency_level >= (n-1))
    Tail.add(t);
  else
    getQueue((t.urgency_level+Head.index)%n).add(t);
}
```

The algorithm recalculates a task's enqueue deadline by subtracting RTT and $l * t$ from its original deadline. Then, it evaluates the task's urgency level. If the task is not able to bear one inter-queue delay, the task is added to queue `Instant`. Otherwise, the task is added to a run queue.

2. $poll(\,)$ This dequeue operation is invoked when an inter-queue delay has expired to move lower level tasks to queue `Instant` using the following algorithm:

```
poll(){
  int to_move = ceil(total_number_of_tasks/n);
  int done = 0;
  for(each Task t in Head){
    Instant.add(Head.remove());
    done++;
  }
  Head = getQueue((Head.index+1)%n);
  int current = Head.index;
  while(done < to_move){
    if(! getQueue(current).isEmpty()){
      Instant.add(getQueue(current).remove());
```

```
        done++;
      }
      else
        current = (current+1)%n;
    }
  }
```

The algorithm firstly moves all tasks in queue `Head` to queue `Instant`. Because each run queue may have a different length, it moves the same number of tasks each time. If less tasks are moved from `Head` to `Instant`, more tasks in lower level queues are moved as well.

6.5.2 Alternative Matchmaking Policies

A matchmaking policy is the set of criteria that an RM uses for selecting locally optimised resource providers (by default, the term "optimised" refers to minimised communication latency). In the Mediator framework, an IMM anticipates forthcoming gaming events in a game zone, and provides in a JAd a list of target players which are likely to interact with a newly respawned NPC. At matchmaking time, an RM takes out every available RAd from its resource queue, matching it to the JobAd using two conditions:

- Does the resource provider have adequate computing resource for running the task?
- Is the mean communication latency among the resource provider and the target players the shortest?

Because this policy strictly selects the resource provider that provides minimum latency, it is called a "strict incentive policy". However, a problem is that less competitive peers are likely to be starved of opportunities for earning credits, and finally to be regarded as

free-riders unfairly, as discussed in Section 6.3.3. To stop this problem from happening, a factor τ can be introduced to relax the selection criteria as below:

- Does the resource provider have adequate computing resource for running the task?
- Is the mean communication latency within the range of $shortest * \tau$?
- Is the resource provider low on credit?

This policy favours the poorest peer providing a communication latency that is not greater than $shortest * \tau$, and making it a "friendly incentive policy". Experimental results demonstrate that with $\tau = 1.2$, DDA shares tasks among the peers more fairly than using the strict incentive policy of $\tau = 1$ (Section 6.7.5).

Actually, DDA allows flexible matchmaking policies to be adopted. For example, when a reputation mechanism is brought into effect in future work, it will allow policies that favour more dependable and trustworthy resource providers.

6.6 Implementation

A proof-of-concept prototype for DDA has been implemented using FreePastry 2.1, an open-source implementation of Pastry [143]. The RM's matchmaking mechanism has been implemented using ClassAds 2.2 [113]. Furthermore, to evaluate the prototype, a test-bed application has been developed, which simulates a P2P MMOG using a 2-dimensional game world and hundreds of virtual player avatars moving according to a random way point algorithm. Figure 6.6 is a UML package diagram that illustrates the key components of the current implementation and their relationships.

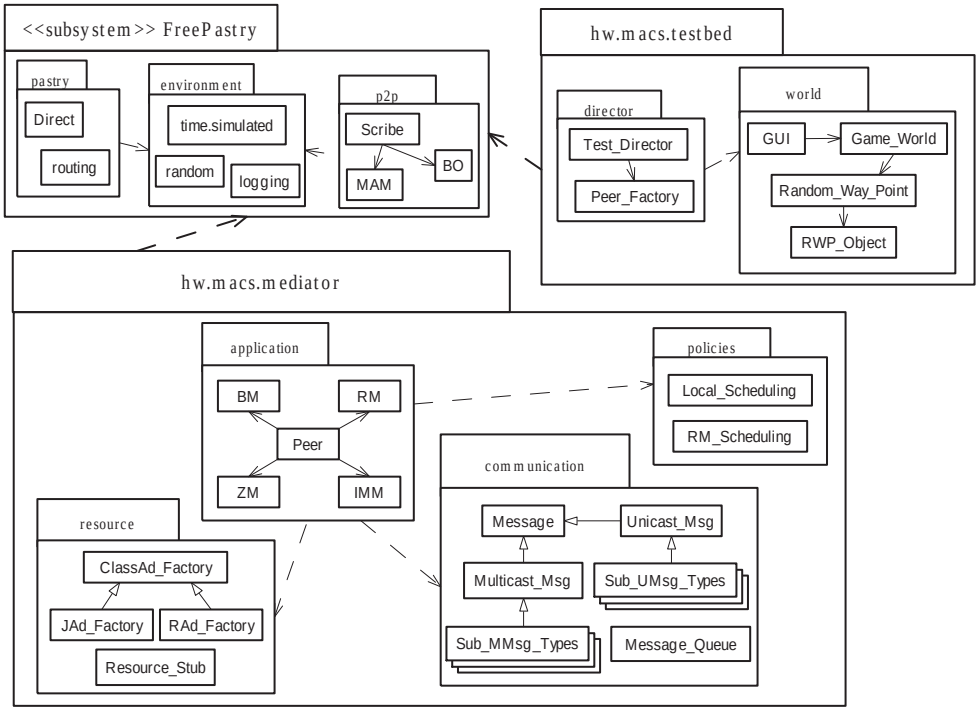

Figure 6.6: Package diagram of the DDA prototype & test bed application

6.6.1 DDA Prototype

In Figure 6.6, the system model of DDA is fulfilled by package `hw.macs.mediator`, which comprises the following sub packages:

1. `hw.macs.mediator.application` This package provides the implementation of a virtual player in a P2P MMOG, together with four super-peer modules that are defined in the Mediator framework (Section 4.3). Each super-peer module is represented by an individual thread, hence when an ordinary player is promoted to a super-peer role, it will activate the corresponding super-peer module by starting a new thread.

In each game zone, the IMM serves as the DDA's work source by constantly generating NPC tasks according to game scenarios. On the other hand, multiple RMs serve as matchmakers that match NPC task requests from the IMM with resource offers from ordinary players.

2. `hw.macs.mediator.communication` The current prototype adopts an event-driven design, so the super-peer modules are essentially event handlers, each being responsible for processing a certain category of messages. The `communication` package defines a set of message types to be used in the prototype. It also provides a `Message Queue` class as a virtual player's communication endpoint. For inbound messages, this class buffers, classifies and dispatches them to corresponding event handlers. The handlers may also schedule outbound messages on this class, which in turn delivers them in the form of unicast, multicast, or anycast messages as required. Last but not least, the package tightly cooperates with the FreePastry subsystem, which supplies message routing functionality in a structured P2P overlay network. Chapter 5 has discussed the MAMBO extension to Scribe ALM middleware. When a super-peer module schedules a multicast message, it can specify whether a MAMBO tree is needed, and if so, what maintenance and bushiness policies should be used.

3. `hw.macs.mediator.resource` This package provides three main functionalities concerning resource representation, discovery and allocation. A `ClassAd Factory` class is implemented as a utility for ordinary players to create resource offers (i.e. RAds) and NPC task requests (i.e. JAds). Secondly, a `Resource Stub` class is implemented to simulate ordinary players' local scheduling activities. Each virtual player is assigned a set of virtual computing resources, based on which the player can decide whether to volunteer for super-peer work, or to request NPC tasks. Thirdly, this package also facilitates the RMs' matchmaking job by determining to what extent an RAd and a JAd are a good match for each other.

4. `hw.macs.mediator.policies` This package contains the flexible policy plug-ins that can be employed by ordinary or super-peers, as indicated by its name. For example, various RM matchmaking policies that are discussed in Section 6.5.2 are collected here.

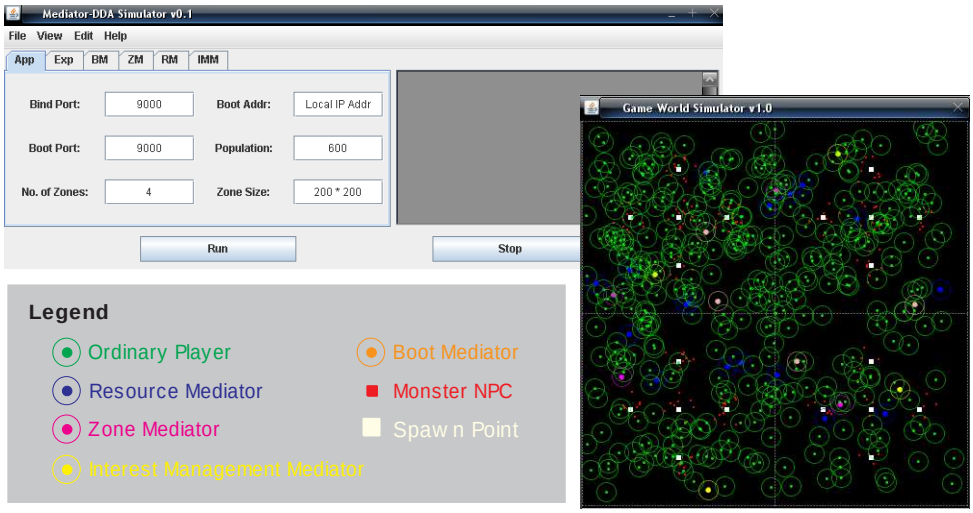

Figure 6.7: Screen shot of the DDA test-bed.

6.6.2 *DDA Test Bed*

To evaluate the DDA prototype a test-bed application was developed, as shown by Figure 6.7. In the screen shot, the GUI for the test-bed is composed of two parts: on the top left is a control panel for adjusting experimental parameters, and on the bottom right, a separate window is provided as the visualizer for a virtual game world. This application is supported by the package `hw.macs.testbed` in Figure 6.6.

1. `hw.macs.testbed.world` The game world is modelled as a 2-dimensional area containing four adjacent game zones. At the beginning of a test, a number of simulated players are created and distributed to random positions in the game world. As players join the game, necessary super-peer roles are selected according to the Mediator framework (Section 4.3), and zone structure information is disseminated to corresponding zone members via ZM heartbeats. The visualizer represents a player's location using a solid point, and the player's AOI as a circle around the point. Different colours are used to reflect a player's super-peer role. After a player has completed its bootstrapping process and settled down in one of the game zones, it starts moving in the game world using the following random way point algorithm:

- Firstly, the player chooses a random destination: $p(x, y) \in [(0, 0), (x_{max}, y_{max})]$, as its next waypoint. The waypoints are uniformly distributed over the game world.
- Secondly, the player chooses a random velocity: $v \in (0, v_{max}]$, which is uniformly distributed as well.
- Thirdly, the player starts moving to the next waypoint p with velocity v.
- Finally, when the player has reached the waypoint, it pauses for a random "thinking time": $t \in [0, t_{max}]$, and then repeats the process.

In each game zone, five "spawn points" are set up for simulating player triggered events. Every time a player in a zone moves towards a spawn point (i.e. the angle between the line connecting the player and the spawn point and the player's moving course is less than $15°$), the IMM anticipates a potential game event. The deadline for that event is calculated as the time that it takes from the player's AOI comes into contact with the spawn point until the player reaches a position that is closest to the spawn point. In addition to such player triggered events, the IMM also respawns a certain amount of monster NPCs periodically, as discussed in Section 6.4. The visualizer represents all the NPC objects as tiny red squares.

2. `hw.macs.testbed.director` This package cooperates with the Direct discrete event simulator integrated in FreePastry, and simulates a whole P2P MMOG with hundreds of virtual players on a single physical machine. The simulator takes in a network topology model generated by GT-ITM [171], and simulates Internet scale communication latencies among peers in a P2P network. The `Peer Factory` class is used to create virtual players, and the `Test Director` class controls at what time a player joins and leaves the application.

6.7 Evaluation

While Section 6.4 presents a theoretical analysis using a mathematical model of the DDA, this section seeks for practical evidence of DDA's effectiveness at allocating NPC hosts for a simulated P2P MMOG. Furthermore, DDA's performance in reducing communication latency for interactive tasks, and establishing a cooperative economic model are compared using four different configurations:

1. Strict Incentive (MFDQ) Matchmakers buffer task requests with a MFDQ, and select locally optimised resource providers using a strict incentive policy.

2. Strict Incentive (EDF) Matchmakers buffer task requests with a EDF queue, and select locally optimised resource providers using a strict incentive policy.

3. Friendly Incentive (1.2) Matchmakers buffer task requests with a MFDQ, and select locally optimised resource providers using a friendly incentive policy, $\tau = 1.2$.

4. Friendly Incentive (2.0) Matchmakers buffer task requests with a MFDQ, and select locally optimised resource providers using a friendly incentive policy, $\tau = 2.0$.

6.7.1 Experimental Environment

Hardware platform The experiments were carried out on a workstation with 2.4GHz Intel Core2 Duo CPU and 2GB memory. Such a hardware platform is able to run simulated P2P MMOG sessions with up to 800 simultaneous players.

Parameter	Meaning	Value
P	zone population	200
Int	respawning interval	6 (second)
TTL	NPC life time expectation	300 (second)
R	NPC:PC ratio	5:1
v_{max}	player's maximal velocity	6 (unit per second)
t_{max}	player's maximal thinking time	10 (seconds)

Table 6.3: DDA test-bed configuration parameters & values

Software setup Table 6.3 lists the parameters and their values used to set up the test-bed application. Firstly, as the maximum number of simultaneous players is 800 and the test-bed application comprises 4 adjacent game zones, the initial population for each zone is set to 200. The IMM's respawning interval is set to 6 seconds, which satisfies Equation 6.6 discussed in Section 6.4. Secondly, the number of periodically respawned NPC tasks in every interval is calculated as $\frac{P*R*Int}{TTL} = \frac{200*5*6}{5*60} = 20$. So, the gaming scenario is simulated to require 20 new NPC objects in every six seconds. Thirdly, for the random way point algorithm, a player's maximal velocity v_{max} and thinking time t_{max} are set to 6 units per second and 10 seconds respectively.

v_{max} has an impact on the deadlines of player-triggered NPC tasks. Figure 6.8 shows the overall deadline distribution of the experiments, where the experimental results indicate that when $v_{max} = 6$, the NPC task deadlines range from 3.0 to 10.5 seconds. On the other hand, t_{max} has an impact on the frequency of player-triggered NPC tasks. Because the deadline for all periodically respawned NPC tasks is the same as the IMM's respawning interval, i.e. 6 seconds, it is the deadline for most of the tasks in Figure 6.8. However, the proportion for tasks having a deadline of 6 seconds never exceeds 50%, which implies that the test-bed application produces more player-triggered tasks than periodically respawned tasks. According to the theoretical analysis in Section 6.4, the number of player-triggered tasks was expected to be $P * r * Int = \frac{200*6}{60} = 20$, which is equal to the number of periodically respawned tasks. Therefore, when $t_{max} = 10$, the

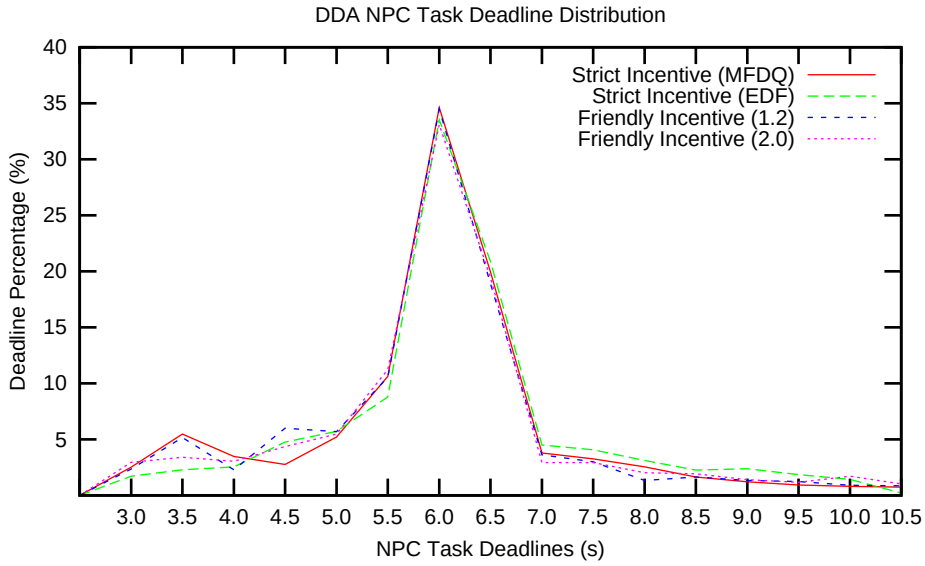

Figure 6.8: DDA NPC task deadline distribution.

test-bed generates a higher amount of mapping workload than the theoretical estimation for a real MMOG. In this case, if DDA is able to support the test-bed effectively, it is likely that DDA is also able to support a real MMOG with an equal number of players.

6.7.2 *Ensuring Comparability of Results*

A challenge for the experiments is to cope with the randomness and to ensure the comparability of experimental results. Above all, because every peer in a P2P overlay network identifies itself with a random PeerId, each time we run the experiment, the topology of the system is different. In addition, due to the random way point model, the initial position of a virtual player in the game world, its moving course and velocity, and the thinking time between each move are also out of control. However, as a coin has two sides, the randomness is helpful from another point of view. For example, it results in various sequences of game events, according to which NPC task requests with different deadlines are generated. Otherwise, it would be difficult to justify the effectiveness of DDA, if all the experiments were repeated with a single, artificial sequence of game events. So, an advisable strategy is to keep the randomness, and to repeat each experi-

ment several times, so as to preserve the commonness and comparability of the results.

In practice, the experiment was repeated 5 times with each of the four configurations. Every time, DDA's performances on reducing communication latency and establishing a cooperative economic model were measured with the first 10,000 tasks. Figure 6.8 indicates that, for equal number of tasks, the overall NPC task deadline distributions of all the configurations are similar to each other. Therefore, further comparisons and analysis that are carried out in Section 6.7.4 and 6.7.5 are meaningful.

6.7.3 DDA's Effectiveness in NPC Host Allocation

DDA's effectiveness in NPC host allocation is investigated with large numbers of application logs collected from the experiments, as demonstrated in Appendix A. As 10,000 tasks are measured in an experiment, and an experiment is repeated 5 times for each of the four configurations, these logs show mapping results for a total number of $10,000 * 5 * 4 = 200,000$ NPC tasks. A mapping is considered to be successful if the IMM has received at least one resource offer from the RMs within the deadline of the corresponding NPC task. In this case, DDA has achieved a 100% success rate for all the 200,000 NPC tasks. So, we draw the conclusion that DDA is able to support a simulated P2P MMOG with 800 players, when NPC task deadlines range from 3.0 to 10.5 seconds. Furthermore, we speculate that DDA can also support a real P2P MMOG effectively, if the scale of the MMOG is similar to our simulation, and the actual gaming scenario and interest management mechanism allow NPC tasks to have similar deadlines.

Because the 200,000 mapping results are collected from experiments with all the four configurations, DDA's extensions do not influence its effectiveness in NPC host allocation. In other words, it does not matter that a resource matchmaker buffers NPC task requests with MFDQ or EDF queue, or it uses what kind of matchmaking policy. DDA is always able to assign large numbers of NPC task to resource providers in due course. However, the use of different extensions may influence how satisfactory the selected

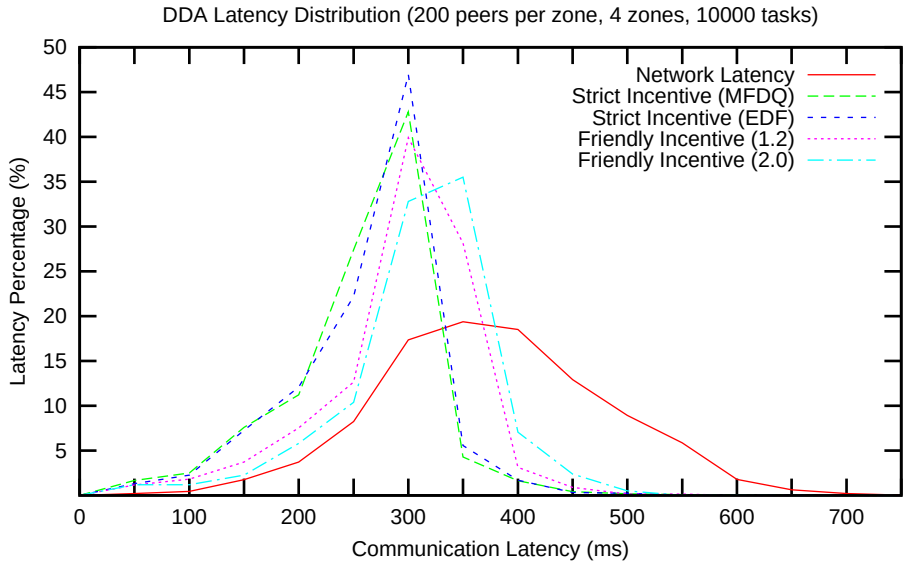

Figure 6.9: DDA latency distribution.

resource providers are, as well as whether NPC tasks are shared among all competent resource providers fairly. These issues are investigated in the following sections.

6.7.4 Communication Latency Optimisation

Figure 6.9 depicts the distribution of communication latencies among resource providers located by DDA, and corresponding resource consumers. Five curves are displayed in this Figure:

1. Network Latency This curve demonstrates the actual communication latency distribution for the GT-ITM network topology used by the Direct simulator. By default, the distribution is approximately a Gaussian distribution $N(350, 100)$. If resource providers are randomly selected in the experiments, the latency distribution for DDA should be similar to this default curve.

2. Strict Incentive (MFDQ) This curve demonstrates DDA's communication latency distribution, when RMs buffer NPC task requests using a MFDQ that is discussed in

Section 6.5.1. Compared to the default network latency distribution, the mean communication latency is reduced by 26.0%, from 350 ms to 259.2 ms. Only around 5% of the NPC hosts are employed, whose communication latencies with target players are higher than the default mean. Accordingly, DDA is able to improve QoS for interactive NPC tasks in a P2P MMOG by carefully allocating NPC hosts to appropriate resource providers.

3. Strict Incentive (EDF) This curve demonstrates DDA's communication latency distribution, when RMs buffer NPC task requests using an intuitive EDF queue. Due to the potential resource tie-up problem (Section 6.5.1), the resource quality attained by the EDF queue is slightly lower than the MFDQ. Concretely, the EDF queue results in a mean communication latency of 265.0 ms, which is 2.3% higher than the 259.2 ms of the MFDQ.

Because the previous comparison of MFDQ and EDF puts much emphasis upon their capabilities of reducing the communication latency among a NPC host and target players, a strict incentive policy is adopted by RMs' matchmaking process. In contrast, the following experiments focus on the effect of the friendly incentive policy, as discussed in Section 6.5.2. Both of them use the same MFDQ, but different τ values are applied to their matchmaking policies.

4. Friendly Incentive (1.2) This curve demonstrates DDA's communication latency distribution, when RMs adopt a friendly incentive policy with a τ value of 1.2. Such a matchmaking policy favours a resource provider that is low on virtual credit and able to guarantee an acceptable QoS, i.e. its communication latency with target players is no longer than $shortest * \tau$. Experimental results show that when $\tau = 1.2$, the mean communication latency attained by DDA is 291.1 ms. This is 12.3% higher than the 259.2 ms of the strict incentive policy, which is equivalent to $\tau = 1$.

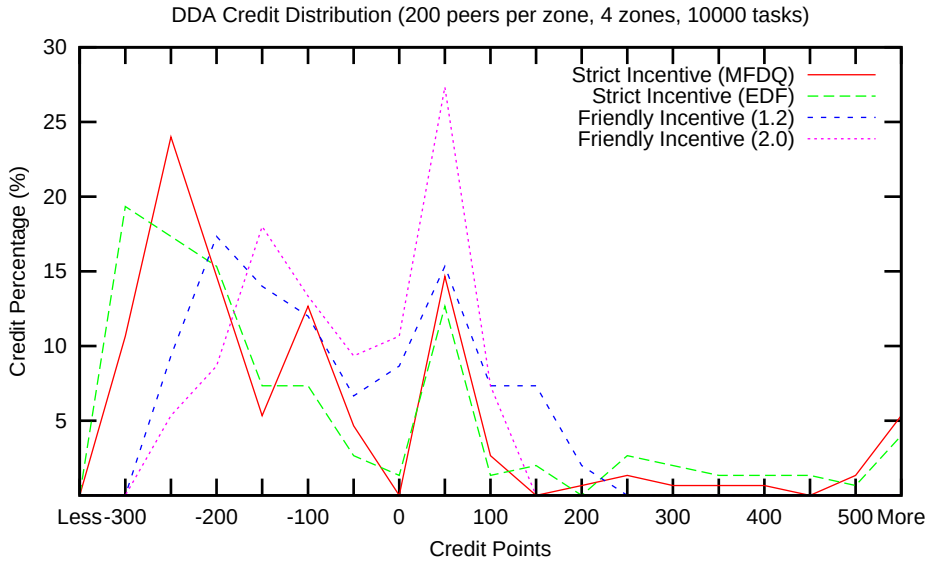

Figure 6.10: DDA credit point distribution.

5. Friendly Incentive (2.0) This curve demonstrates DDA's communication latency distribution, when the τ value is further relaxed to 2.0. As a result, the mean communication latency deteriorates to 307.2 ms, which is 5.5% higher than that of $\tau = 1.2$ and 18.5% higher than that of the strict incentive policy. So, the impact of different τ values on the resource quality is clear - the more relaxed the resource selection criterion is by a friendly incentive policy, the worse the resource quality becomes.

6.7.5 Cooperative Economic Model

Figure 6.10 depicts the distribution of virtual credits earned by the 800 players after 10,000 NPC tasks. No matter whether a MFDQ or an EDF queue is used, with a strict incentive policy, the gap between the richest and poorest players is always wide. Specifically, around 5% of the players have earned more than 500 credits, whereas around 20% other players are in debt to the system for more than -300 credits.

In contrast, with a friendly incentive policy ($\tau = 1.2$), the gap between the rich and poor can be significantly narrowed, as all of the players fall within a range between -300 and

250. It is true that some peers are still in debt, but this is because of their inability to provide usable resources. In the network topology model used by the experiments, some peers suffer from slow network connections and have high communication latencies to other peers in the system. Consequently, although a friendly incentive policy attempts to share credit earning opportunities among application participants fairly, resources from peers that are behind slow connections are seldom employed.

Similarly, when the τ value is more relaxed, e.g. from 1.2 to 2.0, the economic gap can be further narrowed as shown in Figure 6.10. However, due to the negative impact on the resource quality that is discussed in the previous section, it is unadvisable to allow an overly big τ value. In practice, P2P MMOG developers can customise their own economic model to strike a balance between fairness in task sharing and QoS for maintaining acceptable game interactivity.

6.7.6 Evaluation Summary

We make the following key observations from the experimental results:

- DDA efficiently allocates computing resources for large numbers of real-time NPC tasks in a simulated P2P MMOG with 800 simultaneous players (Section 6.7.3).
- DDA supports gaming interactivity by keeping the communication latency among NPC hosts and ordinary players low (Section 6.7.4).
- DDA's MFDQ extension is helpful to address the resource tie-up problem. Compared to an intuitive EDF queue, a MFDQ can increase the probability for DDA to assign NPC tasks to resource providers with higher quality of service (Section 6.7.4).
- DDA's friendly incentive policy extension is helpful to establish a cooperative economic model that shares NPC tasks among competent resource providers fairly.

However, it is inadvisable to allow an overly big τ value, as it relaxes DDA's resource selection criteria and may lower the game interactivity for a MMOG below the level of general acceptability (Section 6.7.5).

6.8 Conclusion

Deadline-Driven Auctions (DDA) is a novel task mapping infrastructure for heterogeneous P2P environments. Its system model comprises a work source, a set of resource providers and a set of matchmakers. The Mediator framework discussed in Chapter 4 fulfils this system model using a self-organising super-peer network, and employs DDA for NPC host allocation purposes (Section 6.3). A theoretical analysis of DDA's design (Section 6.4) shows that DDA can potentially support a P2P MMOG with up to 600 players in each game zone. Two extensions were made to improve DDA's resource discovery and task sharing (Section 6.5). A prototype of DDA and a test-bed application that simulates a P2P MMOG have been implemented (Section 6.6).

The DDA evaluation demonstrates that DDA provides three main advantages. Firstly, it is able to process large numbers of real-time NPC task requests and to support a simulated P2P MMOG with 800 players effectively (Section 6.7.3). Secondly, DDA satisfies the QoS requirement for game interactivity by keeping the communication latency among NPC hosts and ordinary players low, e.g. it reduces the mean network latency by 26.0% (Section 6.7.4). Thirdly, by applying a friendly incentive policy, DDA can establish a cooperative economic model that shares credit earning opportunities among capable application participants fairly (Section 6.7.5).

Compared to previous region and virtual distance based NPC host allocation approaches (Section 3.4), the most significant strength of DDA is that it distributes NPC tasks to all the application participants according to their actual resource availability. In contrast, the region based approach hosts all the NPC objects in a game zone using a single super-peer. Such an approach lacks appropriate load-balancing mechanisms, and thus is not

able to bring the potential of a P2P system into full play. The virtual distance based approach on the other hand, is comparatively better in terms of load-balancing, as it allocates NPC tasks to ordinary game participants. However, this approach does not take into consideration the actual resource availability of a player. Consequently, the player that is closest to a NPC object may have neither adequate computing resources to host it, nor be able to guarantee a low communication latency with other players in the vicinity, which are also likely to interact with that NPC. In contrast, DDA players request NPC tasks according to their local scheduling results, and once a NPC is allocated to a player, the hosting relationship would last until the removal of the NPC or the leaving of the player. Hence DDA also avoids the overhead caused by frequent NPC host switches in the virtual distance based approach.

DDA does, however, have two important limitations. Firstly, DDA is not ideal for 1 to 1 interactions. In some circumstances, a NPC may only need to be present to one player at a time. Such NPCs would be best hosted by a player's own machine, whereas DDA always introduces a third party NPC host, which results in an unnecessary communication overhead. Secondly, DDA's resource matchmaking strategy is opportunistic. In other words, it only attempts to optimise game interactivity for players in the vicinity of an NPC's initial position. However, a NPC may leave the original place where it was spawned, and encounter players from far away. In this case, the communication latency among a NPC host and players that interact with the NPC could be high.

In future work, a way of combining DDA with a virtual distance based approach will be explored so that different NPC host allocation mechanisms can be applied flexibly to optimise both 1:1 and 1:N interactions. Furthermore, DDA's security for local scheduling will be enhanced by a distributed reputation system, so that a new resource matchmaking policy can take into consideration a resource provider's trustworthiness as well as its network latency to target peers and its processing power.

— Learning without thought is labour lost; thought without learning is perilous.

Confucius

Chapter 7

Conclusion

This Chapter discusses the thesis. It starts by summarising the thesis (Section 7.1), before discussing limitations (Section 7.2), outlining ways to progress this research further (Section 7.3), and speculating on the future of P2P MMOGs (Section 7.4). We conclude that ***Mediator*** is a broadly comprehensive and feasible design framework for P2P MMOGs. All the materials presented in this thesis are consolidated by focusing on its findings and achievements.

7.1 Summary

7.1.1 Research Challenges

Traditionally MMOGs have been predominantly implemented as Client/Server systems. This architecture is suitable for many types of distributed applications, and offers advantages such as centralised control, better security and simplicity of implementation. While Client/Server architectures are well suited to MMOGs with small number of players (e.g. 100), MMOGs have been scaling up rapidly since the early 2000s. For example, recently

World of Warcraft [7] has been averaging 10^5 concurrent users [162]. In such a circumstance, substantial computation and communication workload is imposed on the game server infrastructure. To meet these scalability requirements, MMOG service providers have had to expand their game server infrastructures from a single server to multiple servers (Section 2.3). Consequently, the cost of supporting a large scale MMOG with a C/S architecture is considerable [36, 101]. Furthermore, our literature survey (Chapter 2) has also shown that conventional C/S architectures are also confronted with other drawbacks, such as reliability and redundancy.

Peer-to-Peer architectures offer an interesting alternative to conventional C/S architectures (Section 2.4.1). As the processing power and network bandwidth available on personal computers has grown dramatically, it has become possible to migrate a major part of a game server's functionalities to resource-rich client machines to support a MMOG in a P2P fashion at a much lower cost. By exploiting application participants' computing resources, P2P architectures offer potential scalability and reliability benefits (Section 2.5). Moreover, a P2P MMOG could be self-sufficient as long as game participants are willing to contribute their spare resources for both game support and recovery. However, to adapt MMOGs from C/S architectures to P2P architectures completely is a significant research issue, which has given rise to a series of technical challenges (Chapter 3).

7.1.2 A Novel Design Framework for P2P MMOGs

Chapter 4 proposes a broadly comprehensive design framework, called Mediator, that addresses six key issues (Chapter 3) that need to be considered by P2P MMOGs in an integrated system. Mediator uses a self-organising super-peer network that is created on top of a structured P2P overlay. Currently, the framework comprises four super-peer roles, which are BM, ZM, IMM and RM (Section 4.3). The framework is intended to be flexible and extensible, and new super-peer roles can be introduced later on according to newly identified requirements. For example, a set of trusted super-peers could be present in the system for cheating mitigation purposes in the manner of DaCAP [115] or

FreeMMG [40].

The Mediator framework addresses the six key design issues for P2P MMOGs in the following ways:

- **Interest-Management** is carried out by IMM super-peers in each game zone in a MOPAR-like [168] hybrid approach (Section 3.2.3).
- Unicast communication is used for **Event Dissemination** among players that are involved in an interaction.
- Mediator's NPC **Task Sharing** is based on a novel task mapping mechanism for heterogeneous P2P environments, namely Deadline-Driven Auctions (DDA) (Chapter 6).
- The design of DDA provides native support for a DCRC-like [77] **Incentive Mechanism** (Section 6.5.2).
- The Mediator framework aims to provide game **State Persistency** using the PAST [144] distributed storage middleware (Section 4.2.1).
- The design of the Mediator framework allows real-time interactions among players and NPCs to be secured either by proactive **Cheating Mitigation** approaches like NEO [73], SEA [76] and EASES [42], or by reactive approaches like Log Auditing [97], DaCAP [115] and FreeMMG [40] (Section 3.6). Cheating mitigation has been a long-term challenge for all kinds of online games. The Mediator framework is compatible with the latest cheating mitigation technologies, so it has good potential to be able to address cheating issues for P2P MMOGs.

Compared to existing hybrid [139, 108, 91] and fully distributed [59, 86, 57, 79, 118] P2P MMOG architectures, the Mediator framework delivers the following advantages:

- It is designed to distribute the functionalities of traditional game servers to large numbers of game participant machines to achieve efficient resource utilisation. If

a sufficient proportion of game participants are willing to contribute their spare resources to a P2P MMOG, the MMOG could be supported at a very low, or even no extra cost.

- It provides a hierarchical supervision architecture and an efficient membership management mechanism, called MAMBO (Chapter 5), to endow a P2P MMOG with the same level of robustness as a P2P overlay network with limited maintenance overhead.

- Its super-peer and NPC host selection criteria take into consideration participant machines' actual resource availability and heterogeneity. Hence it enables a P2P MMOG to provide smoother gaming experiences.

- It meets inherent challenges for general P2P applications, such as asymmetric network bandwidth, non-dedicated and heterogeneous resources. Therefore, its design is practical and applicable to real Internet environments.

7.1.3 Thesis Achievements

The thesis makes the following research contributions:

- Six key design issues for P2P MMOG architectures have been identified, including interest management, event dissemination, task sharing, state persistency, cheating mitigation and incentive mechanisms. Design alternatives for each issue have been systematically compared, and their interrelationships discussed. Furthermore, the ways in which existing representative P2P MMOG architectures address these issues have been classified, and the completeness of the architectures evaluated (Chapter 3).

- A novel framework for P2P MMOGs, called ***Mediator***, has been designed. This framework distributes the functionalities of conventional game servers to large

numbers of game participant machines, and addresses all the key design issues for P2P MMOGs in an integrated system. Compared with other hybrid and fully distributed P2P MMOG architectures, the Mediator framework provides several advantages in terms of efficient resource utilisation, super-peer selection, fault-tolerance and meeting the inherent challenges for general P2P applications (Chapter 4).

- A novel membership management technology, Membership-Aware Multicast with Bushiness Optimisation (MAMBO), has been designed, implemented and evaluated. MAMBO is primarily designed for the Mediator framework to establish a hierarchical supervision infrastructure that enhances the dependability of its super-peers and NPC hosts. Also, MAMBO can be employed to facilitate membership management in other P2P applications that use tree-based ALMs, such as collaborative computing and multimedia streaming. Evaluation of a demonstration application shows that MAMBO is more scalable than a conventional supervision architecture, and yet incurs less communication overheads (Chapter 5).

- A novel task mapping infrastructure for heterogeneous P2P environments, Deadline-Driven Auctions (DDA), has been designed, analysed, implemented and evaluated. DDA is primarily designed to support NPC host allocation in the Mediator framework, but it can also support the sharing of real-time computational and interactive tasks in other P2P applications, such as distributed video encoding. DDA's working mode as a heterogeneous task mapping infrastructure is different from existing region or virtual distance based NPC host allocation mechanisms. An analysis of a mathematical model of DDA suggests that DDA can support up to 600 players in each game zone, and evaluation of a simulated P2P MMOG further demonstrates that DDA efficiently allocates NPC hosts for approximately 1000 players, as well as keep the communication latency among NPC hosts and ordinary players low. With a friendly matchmaking policy, DDA is also able to share NPC tasks among competent resource providers fairly, and motivate application participants to contribute spare resources to a P2P MMOG (Chapter 6).

7.2 Limitations

The current work has the following limitations:

- Currently only some key components of the Mediator framework have been implemented and evaluated, including super-peer selection, hierarchical supervision, and DDA's task mapping infrastructure for NPC host allocation purposes. Other design components remain to be implemented and evaluated, such as the sub-zoning for super-peers' load-balancing, a real hybrid interest management mechanism for IMMs to anticipate game events and to spawn NPC objects accordingly, and the distributed storage of game states using PAST [144].

- A limitation of the MAMBO membership management technology has been identified but not yet addressed (Section 5.5.6). Because MAMBO distributes supervision responsibilities to forwarder peers in a multicast tree, when a forwarder and its child fail at the same time, the failure of the child cannot be detected and handled. It has been observed from experiments that MAMBO's bushiness optimisation mechanism can increase the probability for simultaneous parent-child failures to occur. A statistical model might account for this issue quantitatively, but such a model has not been built. Furthermore, there are several possible ways of reinforcing MAMBO's supervision architecture, but so far they have not been properly investigated.

- Though MAMBO is primarily designed to enhance the dependability of super-peers and NPC hosts in the Mediator framework, its effectiveness has not been evaluated directly using Mediator's test-bed application (Section 6.6). The test-bed application uses the Direct discrete event simulator provided by FreePastry, an open source implementation of Pastry 2.11. At present the Direct simulator is still under development, and thus it has a few stability problems. In particular, it is vulnerable to the churn effect, when there is a big number of simulated peers,

or the communication workload is high. Unfortunately, the test-bed application is such a program that puts a heavy burden on the simulator. As a result, when some virtual players are intentionally removed from the application to simulate the failures of various super-peers and NPC hosts, the simulator crashes, and thus MAMBO's effectiveness cannot be measured. As an alternative, a simpler demonstration application, P2P Online Market Place (POMP), has been implemented, and MAMBO's performance has been measured with 100 POMP peers. It has allowed inferences to be drawn about MAMBO's effectiveness on the Mediator framework according to the experimental results obtained from POMP.

- The current design of Deadline-Driven Auctions (DDA) has several limitations. Firstly, it results in unnecessary communication latency and overhead for 1 to 1 interactions. In some circumstances, a NPC may only need to be present to one player at a time, and would be best hosted by the player's own machine. However, DDA always introduces a third party NPC host, whereas low communication latency is not optimal compared with no communication latency. Secondly, DDA's opportunistic matchmaking strategy attempts to optimise communication latency for players in the vicinity of a NPC's initial position. However, both NPCs and player avatars are mobile, and it is possible for a NPC to encounter players at a place that is far away from its spawn point. In this case, the communication latency among a NPC host and players that interact with the NPC could be high. Last but not least, as resource providers may cheat on local scheduling, a reputation mechanism that selects dependable and trustworthy resource providers is needed, but has not yet been investigated.

- Finally, as it is infeasible to carry out experiments with a realistic MMOG comprising thousands of computers and people on a real network, a prototype of the Mediator framework was evaluated with a test-bed application that simulates a P2P MMOG using parameter values typical of commercial MMOGs. Furthermore, since real traces for role-playing games are not available, a random way point algorithm was used to control the movements of virtual players in the simu-

lated MMOG. Though many useful experimental results have been obtained using these methods, the design of the Mediator framework and its supporting technologies, such as MAMBO and DDA, would definitely earn more credibility if they had been demonstrated to be able to support a real P2P MMOG.

7.3 Future Work

There are several avenues to extend this research and address both the practical and theoretical limitations identified in the previous section:

To complete the implementation of the Mediator framework: The first avenue is to continue implementing the other components of the Mediator framework. For example, though related work like MOPAR [168] and Meta-Model [140] have demonstrated the practicability of hybrid interest management approaches, it is still necessary to reimplement such a mechanism to make the IMMs fully functional. Furthermore, when a mature version of PAST becomes available in the near future, a prototype of Mediator's game state persistency mechanism should be established, and a series of relevant issues should be studied, including efficient replication schemes [112] to ensure data availability, separation of permanent and ephemeral data [23], and the introduction of various caching, buffering and data holder super-peers [91] to facilitate real-time retrieval and update of game states.

To address the limitations of MAMBO: Another possible avenue is to remove MAMBO's current limitation in handling simultaneous parent-child failures. There are many possible ways of doing this, such as to exploit grandparent and sibling peers in a multicast tree, or to adopt a multi-tree strategy like Splitstream [37]. Comparatively, the multi-tree approach may offer more advantages, as it not only reduces the probability of simultaneous parent-child failures, but also increases the success rate of message delivery.

To address the limitations of DDA: The third avenue to be explored is potential approaches to improve the current design of DDA. Firstly, DDA relies on application participants to carry out local scheduling activities and requests for various computational and interactive tasks. In this case, unscrupulous peers may cheat on local scheduling and lie about the spare computing resource they can provide. For example, a peer may request tasks that are beyond its capability to earn more credits. Therefore, DDA may adopt a distributed reputation system similar to EigenTrust [98] and REPS [89], so that a resource consumer can rate a corresponding resource provider's quality of service, and allow a reputation-aware matchmaking policy to select only dependable and trustworthy resource providers. Secondly, a way of combining DDA with a virtual distance based NPC host allocation mechanism may be possible. Hence different approaches might be applied flexibly to optimise both 1:1 and 1:N interactions. Last but not least, communication latency oriented NPC host handover might be investigated. By default it has been assumed that the same resource provider will host a NPC until the NPC is removed, or the resource provider has to leave a P2P MMOG. However, a NPC might travel some distance in the world and encounter a new group of player characters at a location that is far away from its original spawn point. This NPC may need to be handed over to another host that is able to provide lower communication latency for the latest target players.

To seek new experimental methodologies: Finally, future research might explore new experimental methodologies that enable a simulated MMOG to imitate a real MMOG more closely, so as to obtain more convincing experimental results. For example, instead of using a random way point model, intelligent virtual players might be developed to mimic real players, such as to complete quests, to combat monster NPCs, and to move to different locations in a game world with clear objectives. Presently considerable research efforts have been made to address key design issue for P2P MMOGs, as discussed in Chapter 3. However, they are all confronted with the same challenge, which is the lack of a suitable evaluation platform. Such a platform should be able to simulate representative territory, gaming scenarios, NPC density and distribution, and

various player behaviours in real MMOGs. The provision of such a platform would save researchers the effort of building their own test-bed applications, and more importantly, an identical configuration of the platform could provide performance bench marks for rival P2P MMOG architectures.

7.4 The Future of P2P MMOGs

At the time of writing, all commercial MMOGs like World of Warcraft [7], Ultima Online [14] and EverQuest [5], are implemented as Client/Server systems. In this circumstance, at the end of the thesis the author would like to look forward to the future of P2P MMOGs, and to discuss whether they will really come into existence.

Security - a big challenge for P2P MMOGs Cheating is such a serious problem that MMOG service providers must make an all-out effort to protect their games from any potential attacks and security breaches. A few proactive [26, 73, 76, 42] and reactive [97, 93, 115, 40, 107] cheating mitigation mechanisms for P2P MMOGs have been proposed in the literature (Section 3.6). However, these mechanisms can only prevent a limited number of cheats from happening. In comparison with P2P architectures, C/S architectures are more easily made secure. Hence MMOG service providers have been reluctant to take the risks of trying out alternative architectures to C/S, even though the cost of supporting a MMOG with a centralised game server infrastructure is considerable.

Recently, Trusted Computing (TC), put forward by the Trusted Computing Group [11], has become popular. Two things that TC could provide are of interest to MMOG service providers. First, the possibility that only software that is signed by the producer (and thus is trusted) may run on a TC-enabled computer. Second, the possibility that a TC-enabled computer can prove its trustworthiness to other systems. The first thing would guarantee that the game software could not be manipulated, and the second enable

game providers to identify trusted game participants over the Internet. In this case, a P2P MMOG could safely distribute various computational and administrative tasks to trusted game participants, and even allow players to store account and game state information locally without worrying about the data being tampered with. TC and other similar technologies will supply a gap in the security of distributed applications, and significantly facilitate the deployment of P2P MMOGs.

MMO 2.0 - MMOGs & Web 2.0 are converging MMOGs themselves are evolving rapidly as well. The purpose of MMOGs are changing from pure gaming to a social platform in a broad sense. Jim Crowley, President of Turbine Entertainment, has given an influential keynote talk at Tokyo Game Show '08 about "The Collision of Virtual Worlds, Online Games, and Social Networking" [52]. He referred to the young people that were born from 1995 onwards as the "born digital" generation, who are living their lives publicly in digital space. He pointed out that there is a trend for traditional MMOGs to morph into sophisticated social networks, to serve as a real-time layer of Web 2.0-style [129] social tools, and to provide immersive and directed experiences. In essence, virtual worlds, games and social networks are coming together to form an ultimate community namely "MMO 2.0". At present, SecondLife [9] might be the most noteworthy example of new-style MMOGs. SecondLife demonstrates that the themes and goals of traditional MMOGs have already been changed a lot.

BadumnaSim - the first P2P MMOG is emerging NICTA, Australia's Information and Communications Technology Centre of Excellence, has developed a Badumna network engine to support a MMOG that is similar to SecondLife in a P2P fashion [16]. The technology reduces the cost of maintaining expensive game servers by delegating data processing to individual participants, and can support millions of users with very minimal infrastructure. VastPark, a leading virtual worlds platform provider in Australia, is developing the first P2P MMOG called BadumnaSim. A beta version of the game has been available for testing since 21 October 2008 [122]. Albert Einstein said that

"I never think of the future - it comes soon enough". While we are discussing whether P2P MMOGs are able to come into existence, a commercial P2P MMOG is about to be launched.

Appendix A: DDA Test-Bed Log Sample

Experiment initialisation:

```
29/09 00:32:10 [WARN ] <AWT-EventQueue-0> - 1222558330421 -> Starting a test...
29/09 00:32:10 [WARN ] <AWT-EventQueue-0> - 1222558330437 -> Initializing PeerFactory...
29/09 00:32:10 [WARN ] <AWT-EventQueue-0> - PeerFactory is up.
29/09 00:32:16 [INFO ] <Thread-51 > - Finished creating Peer138: <0xF508B7..>
29/09 00:32:16 [INFO ] <Thread-12 > - Finished creating Peer21: <0x637E4A..>
29/09 00:32:17 [INFO ] <Thread-8  > - Finished creating Peer9: <0x8F64EA..>
29/09 00:32:17 [INFO ] <Thread-13 > - Finished creating Peer24: <0x5F3441..>
29/09 00:32:18 [INFO ] <Thread-35 > - Finished creating Peer90: <0x72E333..>
29/09 00:32:18 [INFO ] <Thread-11 > - Finished creating Peer18: <0x947AB9..>
29/09 00:32:18 [INFO ] <Thread-9  > - Finished creating Peer12: <0x78CC57..>
29/09 00:32:18 [INFO ] <Thread-26 > - Finished creating Peer63: <0x74FF89..>
29/09 00:32:19 [INFO ] <Thread-44 > - Finished creating Peer117: <0x9F1E49..>
29/09 00:32:19 [INFO ] <Thread-6  > - Finished creating Peer3: <0x017FEE..>
29/09 00:32:19 [INFO ] <Thread-30 > - Finished creating Peer75: <0x614F45..>
29/09 00:32:20 [INFO ] <Thread-40 > - Finished creating Peer105: <0xBFE7CF..>
29/09 00:32:21 [INFO ] <Thread-52 > - Finished creating Peer141: <0x6F0675..>
29/09 00:32:21 [INFO ] <Thread-34 > - Finished creating Peer87: <0x7A96EB..>
29/09 00:32:21 [INFO ] <Thread-5  > - Finished creating Peer0: <0x62F89E..>
29/09 00:32:21 [INFO ] <Thread-45 > - Finished creating Peer120: <0xA4F328..>
29/09 00:32:21 [INFO ] <Thread-27 > - Finished creating Peer66: <0x2D2212..>
29/09 00:32:21 [INFO ] <Thread-10 > - Finished creating Peer15: <0x16F432..>
29/09 00:32:21 [INFO ] <Thread-31 > - Finished creating Peer78: <0x4A0E87..>
29/09 00:32:21 [INFO ] <Thread-23 > - Finished creating Peer54: <0xBCE0B5..>
29/09 00:32:21 [INFO ] <Thread-49 > - Finished creating Peer132: <0x8FA55D..>
29/09 00:32:21 [INFO ] <Thread-54 > - Finished creating Peer147: <0x5BACC3..>
29/09 00:32:21 [INFO ] <Thread-33 > - Finished creating Peer84: <0x5B0624..>
29/09 00:32:21 [INFO ] <Thread-50 > - Finished creating Peer135: <0x92F97B..>
29/09 00:32:21 [INFO ] <Thread-38 > - Finished creating Peer99: <0x5C3922..>
29/09 00:32:22 [INFO ] <Thread-20 > - Finished creating Peer45: <0xDE9369..>
29/09 00:32:22 [INFO ] <Thread-46 > - Finished creating Peer123: <0x8BC9F1..>

... ...

29/09 00:33:13 [INFO ] <Thread-22 > - Finished creating Peer752: <0x958B1F..>
29/09 00:33:14 [INFO ] <Thread-18 > - Finished creating Peer741: <0xE595DB..>
29/09 00:33:14 [INFO ] <Thread-19 > - Finished creating Peer744: <0x8968F4..>
29/09 00:33:14 [INFO ] <Thread-24 > - Finished creating Peer758: <0x82E3D7..>
29/09 00:33:14 [INFO ] <Thread-15 > - Finished creating Peer732: <0x070E24..>
29/09 00:33:14 [INFO ] <Thread-29 > - Finished creating Peer774: <0x3FFF8B..>
29/09 00:33:14 [INFO ] <Thread-7  > - Finished creating Peer708: <0xD1CB82..>
29/09 00:33:14 [INFO ] <Thread-41 > - Finished creating Peer710: <0x5A4C18..>
29/09 00:33:14 [INFO ] <Thread-21 > - Finished creating Peer750: <0x0CEC96..>
29/09 00:33:14 [INFO ] <Thread-47 > - Finished creating Peer728: <0xFF2C2C..>
29/09 00:33:14 [INFO ] <Thread-17 > - Finished creating Peer738: <0x1AF10F..>
29/09 00:33:14 [INFO ] <Thread-28 > - Finished creating Peer771: <0xE23293..>
29/09 00:33:14 [INFO ] <Thread-37 > - Finished creating Peer798: <0x6529D1..>
29/09 00:33:14 [INFO ] <Thread-53 > - Finished creating Peer746: <0x84230D..>
29/09 00:33:14 [INFO ] <Thread-39 > - Finished creating Peer704: <0x2E313A..>
29/09 00:33:15 [INFO ] <Thread-48 > - Finished creating Peer731: <0xA0E0C0..>
29/09 00:33:15 [INFO ] <Thread-32 > - Finished creating Peer783: <0xACD7D8..>
29/09 00:33:15 [INFO ] <Thread-22 > - Finished creating Peer753: <0x9F732C..>
29/09 00:33:15 [INFO ] <Thread-24 > - Finished creating Peer759: <0xA9B078..>
29/09 00:33:15 [WARN ] <AWT-EventQueue-0> - 800 peers have been created.
```

Game event anticipation and NPC host allocation:

```
29/09 00:33:16 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 0 on IMM peer6 with deadline: 7605 ms.
29/09 00:33:16 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 1 on IMM peer6 with deadline: 7605 ms.
29/09 00:33:16 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 2 on IMM peer6 with deadline: 4251 ms.
29/09 00:33:16 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 3 on IMM peer6 with deadline: 4251 ms.
29/09 00:33:16 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 4 on IMM peer6 with deadline: 6921 ms.
29/09 00:33:16 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 5 on IMM peer6 with deadline: 6921 ms.
29/09 00:33:16 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 6 on IMM peer6 with deadline: 3983 ms.
29/09 00:33:16 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 7 on IMM peer6 with deadline: 3983 ms.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job2.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job6.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job7.
29/09 00:33:17 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job3.
29/09 00:33:17 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job3.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job5.
29/09 00:33:17 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job0.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job4.
29/09 00:33:17 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job1.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job0.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job1.
29/09 00:33:17 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job2.
29/09 00:33:17 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job6.
29/09 00:33:17 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job7.
29/09 00:33:17 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job3.
29/09 00:33:17 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job5.
29/09 00:33:17 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job4.
29/09 00:33:17 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job6.
29/09 00:33:17 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job7.
29/09 00:33:17 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job0.
29/09 00:33:17 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job1.
29/09 00:33:17 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job4.
29/09 00:33:17 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job5.
29/09 00:33:17 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 8 on IMM peer6 with deadline: 4788 ms.
29/09 00:33:17 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 9 on IMM peer6 with deadline: 4788 ms.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job8.
29/09 00:33:17 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job9.
29/09 00:33:18 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job8.
29/09 00:33:18 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job9.
29/09 00:33:18 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job8.
29/09 00:33:18 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job9.
29/09 00:33:18 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 10 on IMM peer6 with deadline: 4337 ms.
29/09 00:33:18 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 11 on IMM peer6 with deadline: 4337 ms.
29/09 00:33:19 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job10.
29/09 00:33:19 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job11.
29/09 00:33:19 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job10.
29/09 00:33:19 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job10.
29/09 00:33:19 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job11.
29/09 00:33:19 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job11.
29/09 00:33:19 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 12 on IMM peer26 with deadline: 5739 ms.
29/09 00:33:19 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 13 on IMM peer26 with deadline: 5739 ms.
29/09 00:33:20 [INFO ] <Thread-393> - Scheduled general auction 14 on IMM peer6 with deadline: 6000 ms.
29/09 00:33:20 [INFO ] <Thread-394> - Scheduled general auction 15 on IMM peer19 with deadline: 6000 ms.
29/09 00:33:20 [INFO ] <Thread-396> - Scheduled general auction 16 on IMM peer3 with deadline: 6000 ms.
29/09 00:33:20 [INFO ] <Thread-395> - Scheduled general auction 17 on IMM peer26 with deadline: 6000 ms.
29/09 00:33:20 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job12.
29/09 00:33:20 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job13.
29/09 00:33:20 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job12.
29/09 00:33:20 [INFO ] <Thread-393> - Scheduled general auction 18 on IMM peer6 with deadline: 5812 ms.
```

```
29/09 00:33:20 [INFO ] <Thread-394> - Scheduled general auction 19 on IMM peer19 with deadline: 5797 ms.
29/09 00:33:20 [INFO ] <Thread-395> - Scheduled general auction 20 on IMM peer26 with deadline: 5797 ms.
29/09 00:33:20 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job13.
29/09 00:33:20 [INFO ] <Thread-396> - Scheduled general auction 21 on IMM peer3 with deadline: 5797 ms.
29/09 00:33:20 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job12.
29/09 00:33:20 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 22 on IMM peer19 with deadline: 5175 ms.
29/09 00:33:20 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 23 on IMM peer19 with deadline: 5175 ms.
29/09 00:33:20 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job13.
29/09 00:33:20 [INFO ] <Thread-220> - RM Peer16 has located a resource for Job16.
29/09 00:33:20 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job14.
29/09 00:33:20 [INFO ] <Thread-231> - RM Peer7 has located a resource for Job16.
29/09 00:33:20 [INFO ] <Thread-393> - Scheduled general auction 24 on IMM peer6 with deadline: 5609 ms.
29/09 00:33:20 [INFO ] <Thread-394> - Scheduled general auction 25 on IMM peer19 with deadline: 5609 ms.
29/09 00:33:20 [INFO ] <Thread-395> - Scheduled general auction 26 on IMM peer26 with deadline: 5609 ms.
29/09 00:33:20 [INFO ] <Thread-220> - RM Peer16 has located a resource for Job21.
29/09 00:33:20 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job17.
29/09 00:33:20 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job14.
29/09 00:33:20 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job14.
29/09 00:33:20 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job17.
29/09 00:33:20 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job15.
29/09 00:33:20 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job18.
29/09 00:33:20 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job15.
29/09 00:33:20 [INFO ] <Thread-393> - Scheduled general auction 27 on IMM peer6 with deadline: 5406 ms.
29/09 00:33:20 [INFO ] <Thread-231> - RM Peer7 has located a resource for Job21.
29/09 00:33:20 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job17.
29/09 00:33:20 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 28 on IMM peer19 with deadline: 5718 ms.
29/09 00:33:20 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 29 on IMM peer19 with deadline: 5718 ms.
29/09 00:33:20 [INFO ] <pool-2-thread-4> - Auction 6 has completed - provider 25, latency 130, reward 6.
29/09 00:33:20 [INFO ] <pool-2-thread-4> - Auction 7 has completed - provider 277, latency 142, reward 6.
29/09 00:33:20 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job19.
29/09 00:33:20 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job19.
29/09 00:33:20 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job18.
29/09 00:33:20 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job18.
29/09 00:33:20 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job20.
29/09 00:33:20 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job20.
29/09 00:33:20 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job22.
29/09 00:33:20 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job24.
29/09 00:33:20 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job23.
29/09 00:33:21 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job22.
29/09 00:33:21 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job23.
29/09 00:33:21 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job24.
29/09 00:33:21 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job24.
29/09 00:33:21 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job20.
29/09 00:33:21 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job26.
29/09 00:33:21 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job27.
29/09 00:33:21 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job25.
29/09 00:33:21 [INFO ] <pool-2-thread-3> - Auction 2 has completed - provider 45, latency 110, reward 6.
29/09 00:33:21 [INFO ] <pool-2-thread-4> - Auction 3 has completed - provider 147, latency 130, reward 6.
29/09 00:33:21 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job25.
29/09 00:33:21 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job26.
29/09 00:33:21 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job26.
29/09 00:33:21 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job27.
29/09 00:33:21 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job27.
29/09 00:33:21 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job28.
29/09 00:33:21 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job29.
29/09 00:33:21 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job28.
29/09 00:33:21 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job29.
29/09 00:33:22 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 30 on IMM peer6 with deadline: 4347 ms.
29/09 00:33:22 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 31 on IMM peer6 with deadline: 4347 ms.
29/09 00:33:22 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 32 on IMM peer26 with deadline: 5430 ms.
29/09 00:33:22 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 33 on IMM peer26 with deadline: 5430 ms.
```

```
29/09 00:33:22 [INFO ] <pool-2-thread-1> - Auction 8 has completed - provider 618, latency 169, reward 16.
29/09 00:33:22 [INFO ] <pool-2-thread-18> - Auction 9 has completed - provider 542, latency 241, reward 16.

... ...

29/09 00:38:14 [INFO ] <Thread-5441> - Scheduled general auction 2136 on IMM peer6 with deadline: 6000 ms.
29/09 00:38:14 [INFO ] <pool-3-thread-46> - Auction 2113 has completed - provider 113, latency 280, reward 16.
29/09 00:38:14 [INFO ] <Thread-5443> - Scheduled general auction 2137 on IMM peer26 with deadline: 6000 ms.
29/09 00:38:14 [INFO ] <pool-2-thread-37> - Auction 2117 has completed - provider 122, latency 262, reward 21.
29/09 00:38:14 [INFO ] <pool-2-thread-85> - Auction 2109 has completed - provider 18, latency 263, reward 6.
29/09 00:38:14 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job2127.
29/09 00:38:14 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2126.
29/09 00:38:14 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2127.
29/09 00:38:14 [INFO ] <Thread-5441> - Scheduled general auction 2138 on IMM peer6 with deadline: 5797 ms.
29/09 00:38:14 [INFO ] <Thread-220> - RM Peer16 has located a resource for Job2135.
29/09 00:38:14 [INFO ] <Thread-5442> - Scheduled general auction 2139 on IMM peer19 with deadline: 5797 ms.
29/09 00:38:14 [INFO ] <Thread-5444> - Scheduled general auction 2140 on IMM peer3 with deadline: 5797 ms.
29/09 00:38:14 [INFO ] <Thread-5443> - Scheduled general auction 2141 on IMM peer26 with deadline: 5797 ms.
29/09 00:38:14 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job2128.
29/09 00:38:14 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job2128.
29/09 00:38:14 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job2136.
29/09 00:38:14 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job2129.
29/09 00:38:14 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job2129.
29/09 00:38:14 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job2132.
29/09 00:38:14 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job2133.
29/09 00:38:14 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2132.
29/09 00:38:14 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2133.
29/09 00:38:14 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job2130.
29/09 00:38:14 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job2128.
29/09 00:38:14 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job2131.
29/09 00:38:14 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job2129.
29/09 00:38:14 [INFO ] <Thread-5443> - Scheduled general auction 2142 on IMM peer26 with deadline: 5594 ms.
29/09 00:38:14 [INFO ] <Thread-5441> - Scheduled general auction 2143 on IMM peer6 with deadline: 5594 ms.
29/09 00:38:14 [INFO ] <Thread-220> - RM Peer16 has located a resource for Job2140.
29/09 00:38:14 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job2132.
29/09 00:38:14 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job2136.
29/09 00:38:14 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job2133.
29/09 00:38:14 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job2137.
29/09 00:38:14 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job2137.
29/09 00:38:14 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job2130.
29/09 00:38:14 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2136.
29/09 00:38:14 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job2131.
29/09 00:38:14 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job2138.
29/09 00:38:14 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job2134.
29/09 00:38:14 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job2134.
29/09 00:38:14 [INFO ] <Thread-231> - RM Peer7 has located a resource for Job2135.
29/09 00:38:14 [INFO ] <Thread-5441> - Scheduled general auction 2144 on IMM peer6 with deadline: 5391 ms.
29/09 00:38:14 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job2137.
29/09 00:38:14 [INFO ] <Thread-231> - RM Peer7 has located a resource for Job2140.
29/09 00:38:14 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job2141.
29/09 00:38:14 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2138.
29/09 00:38:14 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job2138.
29/09 00:38:14 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job2141.
29/09 00:38:15 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job2143.
29/09 00:38:15 [INFO ] <Thread-240> - RM Peer27 has located a resource for Job2139.
29/09 00:38:15 [INFO ] <Thread-228> - RM Peer11 has located a resource for Job2139.
29/09 00:38:15 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job2141.
29/09 00:38:15 [INFO ] <Thread-237> - RM Peer20 has located a resource for Job2144.
29/09 00:38:15 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job2142.
29/09 00:38:15 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2143.
29/09 00:38:15 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job2143.
```

```
29/09 00:38:15 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job2142.
29/09 00:38:15 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job2142.
29/09 00:38:15 [INFO ] <Thread-234> - RM Peer15 has located a resource for Job2144.
29/09 00:38:15 [INFO ] <Thread-243> - RM Peer104 has located a resource for Job2144.
29/09 00:38:15 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 2145 on IMM peer3 with deadline: 5648 ms.
29/09 00:38:15 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 2146 on IMM peer3 with deadline: 5648 ms.
29/09 00:38:15 [INFO ] <Thread-220> - RM Peer16 has located a resource for Job2145.
29/09 00:38:15 [INFO ] <Thread-220> - RM Peer16 has located a resource for Job2146.
29/09 00:38:15 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 2147 on IMM peer26 with deadline: 5037 ms.
29/09 00:38:15 [INFO ] <AWT-EventQueue-0> - Scheduled dedicated auction 2148 on IMM peer26 with deadline: 5037 ms.
29/09 00:38:16 [INFO ] <pool-1-thread-63> - Auction 2115 has completed - provider 88, latency 306, reward 6.
29/09 00:38:16 [INFO ] <pool-1-thread-29> - Auction 2116 has completed - provider 530, latency 369, reward 6.
29/09 00:38:16 [INFO ] <Thread-231> - RM Peer7 has located a resource for Job2145.
29/09 00:38:16 [INFO ] <Thread-231> - RM Peer7 has located a resource for Job2146.
29/09 00:38:16 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job2147.
29/09 00:38:16 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job2147.
29/09 00:38:16 [INFO ] <Thread-217> - RM Peer9 has located a resource for Job2148.
29/09 00:38:16 [INFO ] <Thread-246> - RM Peer141 has located a resource for Job2148.
29/09 00:38:16 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job2147.
29/09 00:38:16 [INFO ] <Thread-225> - RM Peer10 has located a resource for Job2148.
29/09 00:38:16 [INFO ] <pool-4-thread-16> - Auction 2124 has completed - provider 362, latency 232, reward 21.
29/09 00:38:16 [INFO ] <pool-4-thread-23> - Auction 2125 has completed - provider 77, latency 233, reward 11.
```

Billing results:

```
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer0's final credit - 234.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer1's final credit - -294.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer2's final credit - -256.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer3's final credit - -252.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer4's final credit - 222.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer5's final credit - -87.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer6's final credit - 78.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer7's final credit - 56.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer8's final credit - -213.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer9's final credit - 71.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer10's final credit - 43.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer11's final credit - 397.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer12's final credit - -101.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer13's final credit - -201.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer14's final credit - -225.
29/09 00:38:16 [INFO ] <AWT-EventQueue-0> - Peer15's final credit - 19.

... ...

29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer785's final credit - 11.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer786's final credit - -284.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer787's final credit - 54.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer788's final credit - -107.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer789's final credit - -125.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer790's final credit - -284.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer791's final credit - 20.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer792's final credit - 300.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer793's final credit - 1292.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer794's final credit - -256.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer795's final credit - -172.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer796's final credit - -213.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer797's final credit - 63.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer798's final credit - -255.
29/09 00:38:17 [INFO ] <AWT-EventQueue-0> - Peer799's final credit - -173.
```

Appendix B: Log Sample Analysis

Communication latency distribution:

```
#NPC host allocation failures: 0.0%
#NPC hosts with unknown communication latency: 0.4%
#NPC hosts with specific communication latency:
0 ~ 49 ms: 0.0%
50 ~ 99 ms: 1.3%
100 ~ 149 ms: 2.3%
150 ~ 199 ms: 7.3%
200 ~ 249 ms: 12.1%
250 ~ 299 ms: 22.2%
300 ~ 349 ms: 47.0%
350 ~ 399 ms: 5.6%
400 ~ 449 ms: 1.7%
450 ~ 499 ms: 0.3%
500 ~ 549 ms: 0.2%
550 ~ 599 ms: 0.0%
600 ms & more: 0.0%
```

Virtual credit distribution:

```
-400 & less: 0.0%
-399 ~ -350: 0.0%
-349 ~ -300: 10.6%
-299 ~ -250: 24.0%
-249 ~ -200: 14.8%
-199 ~ -150: 5.6%
-149 ~ -100: 12.4%
-99 ~ -50: 4.2%
-49 ~ 0: 0.0%
1 ~ 50: 14.8%
51 ~ 100: 2.8%
101 ~ 150: 0.2%
151 ~ 200: 0.8%
201 ~ 250: 1.2%
251 ~ 300: 0.6%
301 ~ 350: 0.6%
351 ~ 400: 0.6%
401 ~ 450: 0.0%
451 ~ 500: 1.6%
501 and more: 5.2%
```

Bibliography

[1] British Legends. british-legends.com, Richard Bartle & Co., 1976.

[2] IEEE Standard 1278 - Protocols for Distributed Interactive Simulation Applications. *IEEE Standard for Information Technology*, May 1993.

[3] Announcement of Weakness in the Secure Hash Standard. Technical report, National Institute of Standards and Technology (NIST), May 1994.

[4] The Microsoft Network. msn.com, Microsoft, 1995.

[5] EverQuest. eqplayers.station.sony.com, Sony Online Entertainment, 1999.

[6] IEEE Standard 1516 - High Level Architecture (HLA) Framework and Rules. *IEEE Standard for Modeling and Simulation*, 2000.

[7] World of Warcraft. worldofwarcraft.com, Blizzard Entertainment, 2001.

[8] Butterfly.net: Powering Next-generation Gaming with On-demand Computing. e-business case study, International Data Corporation, www.idc.com, 2002.

[9] Second Life. secondlife.com, Linden Labs, 2003.

[10] World of Legend. woool.sdo.com, Shanda Entertainment, 2004.

[11] Trusted computing group. Technical report, www.trustedcomputinggroup.org, 2005.

[12] Overview of Virtual Environments. Technical report, virtualenvironments.info, 2007.

[13] The Sims Online. thesimsonline.com, Electronic Arts, 2007.

[14] Ultima Online. uo.com, Electronic Arts, 2007.

[15] Blue Falcon Networks - Company Profiles & Financials. Technical report, Hoover's Company Records, 2008.

[16] NICTA Slashes Hosting Costs for MMO P2P Networks. Technical report, www.itnews.com.au, October 2008.

[17] D. T. Ahmed and S. Shirmohammadi. An Expedite State Dissemination Mechanism for MMOGs. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks (i-span)*, pages 199–203. IEEE Computer Society, May 2008.

[18] S. Alexa. America's Army Game: Its (Virtual) Reality Representation and Cocaine. In *Proceedings of the International Conference on Cyberworlds (CW)*, pages 432–438. IEEE, 2004.

[19] T. Alexander. *Massively Multiplayer Game Development*. Charles River Media, ISBN: 1584503904, 2nd edition, 2005.

[20] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: An Experiment in Public-Resource Computing. *Communications of the ACM*, 45(11):56–61, November 2002.

[21] T. W. Anderson. *An Introduction to Multivariate Analysis 2nd Edition*. John Wiley & Sons, 1984.

[22] J. Armstrong. *Making Reliable Distributed Systems in the Presence of Software Errors*. PhD thesis, Royal Institute of Technology, Swedish Institute of Computer Science, December 2003.

[23] M. Assiotis and V. Tzanov. A Distributed Architecture for MMORPG. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*. ACM, 2006.

[24] M. Baker. Cluster computing white paper. *The Computing Research Repository (CoRR)*, cs.DC/0004014:1–119, April 2000.

[25] L. A. Barroso, J. Dean, and U. Hölzle. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 23:22–28, 2003.

[26] N. E. Baughman, M. Liberatore, and B. N. Levine. Cheat-Proof Playout for Centralized and Peer-to-Peer Gaming. *IEEE/ACM Transactions on Networking (TON)*, 15(1):1–13, 2007.

[27] S. Benford and L. E. Fahlen. A Spatial Model of Interaction in Large Virtual Environments. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW)*, pages 107–123. IEEE, 1993.

[28] C. Bengt and G. Rune. The Rise and Fall of Napster - An Evolutionary Approach. In *Proceedings of the 6th International Computer Science Conference on Active Media Technology (ICAMT)*, 2001.

[29] A. Bharambe. Colyseus: A Distributed Architecture for Online Multiplayer Games. In *Proceedings of Symposium on Networked Systems Design and Implementation (NSDI)*, pages 3–6. USENIX, 2006.

[30] J. Blascovich, J. Loomis, A. Beall, K. Swinth, C. Hoyt, and J. Bailenson. Immersive Virtual Environment Technology as a Methodological Tool for Social Psychology. *Psychological Inquiry*, 13(2):103–124, 2002.

[31] J.-S. Boulanger, J. Kienzle, and C. Verbrugge. Comparing Interest Management Algorithms for Massively Multiplayer Games. In *Proceedings of 5th SIGCOMM Workshop on Network and System Support for Games (NetGames)*. ACM, 2006.

[32] T. D. Braun, H. J. Siegel, and A. A. Maciejewski. Heterogeneous Computing: Goals, Methods, and Open Problems. In *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 1–12. CSREA Press, June 2001.

[33] D. Brookshier, D. Govoni, N. Krishnan, and J. C. Soto. *JXTA: Java P2P Programming*. Sams Publishing, ISBN: 0672323664, March 2002.

[34] E. Buyukkaya and M. Abdallah. Data Management in Voronoi-Based P2P Gaming. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC)*, pages 1050–1053. IEEE, 2008.

[35] E. Callaway. The shape of protein structures to come. *Nature*, 449:765, October 2007.

[36] A. Carpenter. Applying Risk Analysis To Play-Balance RPGs. Technical report, Gamasutra.com, 2003.

[37] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, and A. Rowstron. Splitstream: High-bandwidth Multicast in Cooperative Environments. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*. ACM, 2003.

[38] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A Large-scale and Decentralized Application-level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1489–1499, 2002.

[39] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. Scalable Application-Level Anycast for Highly Dynamic Groups. In *Proceedings of Networked Group Communication (NGC)*, October 2003.

[40] F. R. Cecin, R. Real, R. de Oliveira Jannone, C. F. R. Geyer, M. G. Martins, and J. L. V. Barbosa. FreeMMG: A Scalable and Cheat-Resistant Distribution Model for Internet Games. In *Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT)*, pages 83–90. IEEE Computer Society, 2004.

[41] C. Chambers, W. chang Feng, W. chi Feng, and D. Saha. Mitigating Information Exposure to Cheaters in Real-Time Strategy Games. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 7–12. ACM, 2005.

[42] M.-C. Chan, S.-Y. Hu, and J.-R. Jiang. An Efficient and Secure Event Signature (EASES) Protocol for Peer-to-Peer Massively Multiplayer Online Games. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 52(9):1838–1845, June 2008.

[43] T. D. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM (JACM), ISSN:0004-5411*, 43(2):225–267, March 1996.

[44] X. Changqiao, G. Muntean, E. Fallon, and A. Hanley. A Balanced Tree-Based Strategy for Unstructured Media Distribution in P2P Networks. In *Proceedings of International Conference on Communications (ICC)*, pages 1797–1801. IEEE, May 2008.

[45] B. D. Chen and M. Maheswaran. A Fair Synchronization Protocol with Cheat Proofing for Decentralized Online Multiplayer Games. In *Proceedings of the Network Computing and Applications, Third IEEE International Symposium (NCA)*, pages 372–375. IEEE Computer Society, 2004.

[46] F. Chen and V. Kalogeraki. Adaptive Real-Time Update Dissemination in Distributed Virtual Simulation Environments. In *Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 233–236. IEEE Computer Society, 2005.

[47] J. Chen and B. Wu. Locality Aware Dynamic Load Management for Massively Multiplayer Games. In *Proceeding of Principles and Practice of Parallel Programming (PPoPP)*, pages 289–300. IEEE, 2005.

[48] J. Cheng, C. Yan, K. Yu, and J. Ma. A New Live Streaming Media Architecture for Peer-to-Peer Network. In *Proceedings of International Conference on Multimedia and Ubiquitous Engineering (MUE)*, pages 373–377. IEEE, April 2008.

[49] C. J. Cohen, R. C. Buse, D. Haanpaa, and C. J. Jacobus. From HLA to MMOG and Back Again. Technical report, Simulation Interoperability Standards Organization, www.sisostds.org, 2004.

[50] G. Conn, C. Doctorow, and J. Henson. Collaboration Object Lookup Architecture. Technical report, opencola.com.

[51] E. Cronin, B. Filstrup, A. R.Kurc, and S. Jamin. An Efficient Synchronization Mechanism for Mirrored Game Architectures. In *Proceedings of the 1st SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 7–30. ACM, 2002.

[52] J. Crowley. The Collision of Virtual Worlds, Online Games, and Social Networking. Keynote of tokyo game show (tgs '08), Turbine Entertainment, October 2008.

[53] A. Davison. *Killer Game Programming in Java*. O'Reilly, ISBN: 0596007302, 2005.

[54] J.-F. de Buren. Sun Game Server Technology White Paper. Technical report, Sun, 2004.

[55] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. RFC 2903 - Generic AAA Architecture. Technical report, ISOC Network Working Group, August 2000.

[56] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, 1990.

[57] C. G. Dickey, D. Zappala, and V. Lo. A Fully Distributed Architecture for Massively Multiplayer Online Games. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, page 171. ACM, 2004.

[58] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network*, 14:78–88, 2000.

[59] S. Douglas, E. Tanin, and A. Harwood. Enabling Massively Multi-Player Online Gaming Applications on a P2P Architecture. In *Proceedings of the IEEE International Conference on Information and Automation (ICIA)*, pages 7–12. IEEE, 2005.

[60] D. Doval and D. O'Mahony. Overlay Networks - a Scalable Alternative for P2P. *IEEE Internet Computing*, July 2003.

[61] N. Ducheneaut and R. J.Moore. The Social Side of Gaming: a Study of Interaction Patterns in a Massively Multiplayer Online Game. In *Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 360–369. ACM, 2004.

[62] T. N. B. Duong and S. Zhou. A Dynamic Load Sharing Algorithm for Massively Multiplayer Online Games. In *Proceeding of the 11th IEEE International Conference on Networks (ICON)*, pages 131–136. IEEE, 2003.

[63] A. El-Sayed, V. Roca, and L. Mathy. A Survey of Proposals for an Alternative Group Communication Service. *IEEE Network*, 17(1):46–51, January 2003.

[64] L. Fan, H. Taylor, and P. Trinder. Mediator: A Design Framework for P2P MMOGs. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 43–48. ACM, September 2007.

[65] L. Fan, P. Trinder, and H. Taylor. MAMBO: Membership-Aware Multicast with Bushiness Optimisation. In *Short Papers for the 2nd International Conference on Distributed Event-Based Systems (DEBS)*, Rome, Italy, July 2008. DEBS.

[66] L. Fan, P. Trinder, and H. Taylor. Deadline-Driven Auctions for NPC Host Allocation in P2P MMOGs. In *Proceedings of 2nd International Workshop on Massively Multiuser Virtual Environments at IEEE Virtual Reality (MMVE)*, pages 1–7. IEEE, March 2009.

[67] L. Fan, P. Trinder, and H. Taylor. Design Issues for Peer-to-Peer Massively Multiplayer Online Games. In *Proceedings of 2nd International Workshop on Massively Multiuser Virtual Environments at IEEE Virtual Reality (MMVE)*, pages 1–11. IEEE, March 2009.

[68] S. Fiedler, M. Wallner, and M. Weber. A Communication Architecture for Massive Multiplayer Games. In *Proceedings of the 1st SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 14–22. ACM, 2002.

[69] R. Fine. MMOG Considerations. Technical report, www.gamedev.net, 2004.

[70] R. F. Freund and H. J. Siegel. Heterogeneous Processing. *IEEE Computer*, 26(6):13–17, June 1993.

[71] Y. S. Fung. Hack-Proof Synchronization Protocol for Multi-Player Online Games. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*. ACM, 2006.

[72] C. GauthierDickey, V. Lo, and D. Zappala. Using N-Trees for Scalable Event Ordering in Peer-to-Peer Games. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 87–92. ACM, 2005.

[73] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr. Low Latency and Cheat-Proof Event Ordering for Peer-to-Peer Games. In *Proceedings of the 14th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 134–139. ACM, 2004.

[74] M. Gillen, K. Rohloff, P. Manghwani, and R. Schantz. Scalable, Adaptive, Time-Bounded Node Failure Detection. In *Proceedings of the 10th High Assurance Systems Engineering Symposium (HASE '07)*, pages 179–186. IEEE, November 2007.

[75] Y. Goldfinger, S. Vigiser, A. Amir, A. Vardi, and Y. Vardi. ICQ Instant Messaging Computer Program. Technical report, icq.com, America Online, 1996.

[76] M. Gorawski and K. Stachurski. A Secure Event Agreement (SEA) Protocol for Peer-to-Peer Games. In *Proceedings of the First International Conference on Availability, Reliability and Security (ARES)*, pages 34–41. IEEE Computer Society, 2006.

[77] M. Gupta, P. Judge, and M. Ammar. A Reputation System for Peer-to-Peer Networks. In *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 144–152. ACM, 2003.

[78] E. Hammami. Towards a Peer-to-Peer Content Discovery and Delivery Architecture for Service Provisioning. In *Proceedings of Fourth European Conference on Universal Multiservice Networks (ECUMN '07)*, pages 52–61. IEEE, Feb. 2007.

[79] T. Hampel, T. Bopp, and R. Hinn. A Peer-to-Peer Architecture for Massive Multiplayer Online Games. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*. ACM, 2006.

[80] W. Harvey and J. Ventrella. There. Technical report, there.com, Makena Technologies, 2003.

[81] D. Hausheer and B. Stiller. PeerMart: the Technology for a Distributed Auction-Based Market for Peer-to-Peer Services. In *Proceedings of International Conference on Communications (ICC)*, pages 1583– 1587. IEEE, May 2005.

[82] M. Hefeeda and B. Bhargava. On-Demand Media Streaming Over the Internet. In *Proceedings of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS)*, pages 279–285. IEEE, May 2003.

[83] O. Herrera and T. Znati. Modeling Churn in P2P Networks. In *Proceedings of the 40th Annual Simulation Symposium (ANSS)*, pages 33–40. IEEE, March 2007.

[84] M. Hosseini, D. Ahmed, S. Shirmohammadi, and N. Georganas. A Survey of Application-Layer Multicast Protocols. *Communications Surveys and Tutorials, IEEE*, 9(3):58–74, 2007.

[85] M. M. Hu and B. Chang. Massively Multiplayer Online Game Supported Foreign Language Listening Ability Training. In *Proceedings of the The First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL)*, pages 176–178. IEEE Computer Society, 2007.

[86] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang. Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC)*, pages 1134–1138. IEEE, 2008.

[87] 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, July 2006.

[88] S.-Y. Hu and G.-M. Liao. Scalable Peer-to-Peer Networked Virtual Environment. In *Proceedings of the 3rd SIGCOMM Workshop on Network and System Supports for Games (NetGames)*, pages 129–133. ACM, 2004.

[89] G.-Y. Huang, S.-Y. Hu, and J.-R. Jiang. Scalable Reputation Management for P2P MMOGs. In *Proceedings of IEEE Virtual Reality Workshop on Massively Multiuser Virtual Environment (MMVE)*. IEEE, March 2008.

[90] O. H. Ibarra and C. E. Kim. Heuristic Algorithms for Scheduling Independent Tasks on Non-Identical Processors. *Journal of the ACM*, 24(2):280–289, April 1977.

[91] T. Iimura, H. Hazeyama, and Y. Kadobayashi. Zoned Federation of Game Servers - a Peer-to-peer Approach. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 116–120. ACM, 2004.

[92] G. Inc. Form S-1 Registration Statement. *Securities & Exchange Commission, Washington, D.C. 20549*, 2004.

[93] T. Izaiku, S. Yamamoto, Y. Murata, N. Shibata, K. Yasumoto, and M. Ito. Cheat Detection for MMORPG on P2P Environments. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*. ACM, 2006.

[94] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garc'es-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *Proceedings of the 5th Passive and Active Measurement Workshop (PAM)*. Springer, 2004.

[95] D. James and G. Walton. 2004 Persistent Worlds Whitepaper. Technical report, IGDA Online Games SIG, www.igda.org, 2004.

[96] C. J.Bonk and V. P.Dennen. Massive Multiplayer Online Gaming: A Research Framework for Military Training and Education. Technical report, Advanced Distributed Learning (ADL), 2005.

[97] P. Kabus, W. W. Terpstra, M. Cilia, and A. P. Buchmann. Addressing Cheating in Distributed MMOGs. In *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 1–6. ACM, 2005.

[98] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, pages 640–651. ACM, 2003.

[99] K. Kant, R. Iyer, and V. Tewari. A Framework for Classifying Peer-to-Peer Technologies. In *Proceedings of the 2nd International Symposium on Cluster Computing and the Grid (CCGrid)*, page 368. IEEE, 2002.

[100] P. Karbhari, M. Ammar, A. Dhamdhere, H. Raj, G. Riley, and E. len Zegura. Bootstrapping in Gnutella: A Measurement Study. In *Proceedings of the 5th Passive and Active Measurement Workshop (PAM)*. Springer, 2004.

[101] J. Kesselman. Server Architectures for Massively Multiplayer Online Games. In *Session TS-1084 Javaone Conference*. Sun, 2005.

[102] J.-K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. D. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, R. Joshi, A. Kaul, A. Sharma, S. Sripada, P. Vangari, and S. S. Yellampalli. Dynamic Mapping in a Heterogeneous Environment with Tasks Having Priorities and Multiple Deadlines. *Journal of Parallel and Distributed Computing*, 67(2):154–169, 2007.

[103] C. Klaus and G. Frame. Kaneva. Technical report, kaneva.com, Kaneva Corporation, 2004.

[104] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 190–201. ACM, November 2000.

[105] F. Kuhl, R. Weatherly, and J. Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall PTR, ISBN: 0130225115, October 1999.

[106] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. A. Huberman. Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. *Multiagent and Grid Systems*, 1(3), August 2005.

[107] P. Laurens, R. F. Paige, P. J. Brooke, and H. Chivers. A Novel Approach to the Detection of Cheating in Multiplayer Online Games. In *Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS)*, pages 97–106. IEEE Computer Society, 2007.

[108] H.-H. Lee and C.-H. Sun. Load-Balancing for Peer-to-Peer Networked Virtual Environment. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*. ACM, 2006.

[109] E. Léty, T. Turletti, and F. Baccelli. SCORE: A Scalable Communication Protocol for Large-Scale Virtual Environments. *IEEE/ACM Transactions on Networking (TON)*, 12(2):247–260, April 2004.

[110] H. C. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. Bar Gossip. In *Proceedings of the 7th Symposium on Operating System Design and Implementation (OSDI)*, pages 191–204. ACM, November 2006.

[111] Z. Li, G. Xie, Z. Li, Y. Zhang, and X. Duan. DHT-Aid, Gossip-Based Heterogeneous Peer-to-Peer Membership Management. In *Proceedings of the 5th Consumer Communications and Networking Conference (CCNC)*, pages 284–288. IEEE, January 2008.

[112] Y. Lin, B. Kemme, M. Patino-Martinez, and R. Jimenez-Peris. Applying Database Replication to Multi-Player Online Games. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*. ACM, 2006.

[113] M. Litzkow, M. Livny, and M. Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems (ICDCS)*, pages 104–111, June 1988.

[114] D. Liu, L. Vo, and J. Gainsbrugh. Gaia Online. Technical report, gaiaonline.com, Gaia Interactive, 2003.

[115] H.-I. Liu and Y.-T. Lo. DaCAP - A Distributed Anti-Cheating Peer to Peer Architecture for Massive Multiplayer On-line Role Playing Game. In *Proceedings of the Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 584–589. IEEE Computer Society, 2008.

[116] S. Liu, J. Li, and X. Wang. Local Reputation for P2P MMOG Design. In *Proceedings of the Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 523–528. IEEE Computer Society, 2007.

[117] V. Lo, D. Zhou, Y. Liu, C. G. Dickey, and J. Li. Scalable Supernode Selection in Peer-to-Peer Overlay Networks. In *Proceeding of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems (IPTPS)*, pages 18–25. IEEE, 2005.

[118] H. Lu, B. Knutsson, W. Xu, and B. Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. In *Proceeding of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 7–11. IEEE, 2004.

[119] M. Macedonia, M. Zyda, D. Pratt, P. Barham, and S. Zeswitz. NPSNET: A Network Software Architecture for Large Scale Virtual Environments. *Presence*, 3(4):265–287, 1994.

[120] N. Matsumoto, Y. Kawahara, H. Morikawa, and T. Aoyama. A Scalable and Low Delay Communication Scheme for Networked Virtual Environments. In *Proceedings of Global Telecommunications Conference (GLOBECOM)*, pages 529–535. IEEE, 2005.

[121] M. Merabti and A. E. Rhalibi. Peer-to-peer Architecture and Protocol for a Massively Multiplayer Online Game. In *Proceedings of the Global Telecommunications Conference Workshops (GlobeCom)*, pages 519–528. IEEE, 2004.

[122] K. Mills. NICTA and VastPark Enter into Licence Agreement for NICTA's Distributed Virtual Worlds Technology. Technical report, NICTA, March 2008.

[123] A. Mislove, A. Post, A. Haeberlen, and P. Druschel. Experiences in Building and Operating ePost, a Reliable Peer-to-Peer Application. In *Proceedings of European Professional Society for Systems conference (EuroSys)*, pages 147–159. ACM, October 2006.

[124] A. Montresor. A Robust Protocol for Building Superpeer Overlay Topologies. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P)*, pages 202–209. IEEE, 2004.

[125] G. Morgan, F. Lu, and K. Storey. Interest Management Middleware for Networked Games. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (i3D)*. ACM, 2005.

[126] J. Muller, A. Gossling, and S. Gorlatch. On Correctness of Scalable Multi-Server State Replication in Online Games. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 1–11. ACM, 2006.

[127] J. Mulligan and B. Patrovsky. *Developing Online Games - An Insiders Guide*. New Riders Publishing, ISBN: 1592730000, 2003.

[128] D. Nicolas, Y. Nicholas, N. Eric, and M. R. J. "Alone together?": Exploring the Social Dynamics of Massively Multiplayer Online Games. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 407–416. ACM Press, 2006.

[129] T. O'Reilly. What Is Web 2.0. Technical report, O'Reilly Media Inc., September 2005.

[130] V. Pai, K. Tamilmani, V. Sambamurthy, K. Kumar, and A. Mohr. Chainsaw: Eliminating Trees from Overlay Multicast. In *Proceedings of International Workshop on Peer-To-Peer Systems (IPTPS)*, pages 127–140, 2005.

[131] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the ACM SIGMOD International Conference on Management of Data, ISBN:0-89791-268-3*, pages 109–116, Chicago, Illinois, United States, 1988. ACM.

[132] J. U. Putri, S. Wati, V. Evania, and I. W. S. W. Multiple Super Peers to Reduce Single Failure of Super Peer. In *Proceedings of International Conference on Information Integration and Web-based Application and Services (iiWAS)*, pages 491–496, Jakarta, December 2007. Austrian Computer Society.

[133] R. Raman, M. Livny, and M. Solomon. Resource Management through Multilateral Matchmaking. In *Proceedings of the 9th IEEE Symposium on High Performance Distributed Computing (HPDC)*, pages 290–291, Pittsburgh, Pennsylvania, August 2000. IEEE.

[134] S. Ratnasamy, P. Francis, M. Handly, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, pages 161–172. ACM, 2001.

[135] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-Level Multicast Using Content-Addressable Networks. In *Proceedings of the 3rd International Workshop on Networked Group Communication (NGC)*, pages 14–29, November 2001.

[136] A. R.Bharambe, S. Rao, and S. Seshan. Mercury: A Scalable Publish-Subscribe System for Internet Games. In *Proceedings of the 1st SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 3–9. ACM, 2002.

[137] O. Regev and N. Nisan. The POPCORN Market - An Online Market for Computational Resources. In *Proceedings of the First International Conference on Information and Computation Economies (ICE)*, pages 148–157, Charleston, South Carolina, United States, 1998. ACM.

[138] M. Reynal. A Short Introduction to Failure Detectors for Asynchronous Distributed Systems. *ACM SIGACT News*, 36(1):53–70, March 2005.

[139] A. E. Rhalibi and M. Merabti. Agents-based Modeling for a Peer-to-Peer MMOG Architecture. *Computers in Entertainment*, 3(2):3, 2005.

[140] A. E. Rhalibi and M. Merabti. Interest Management and Scalability Issues in P2P MMOG. In *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC)*, pages 1188–1192. IEEE, 2006.

[141] A. Robertson. Linux-HA Heartbeat System Design. In *Proceedings of the 4th annual Linux Showcase & Conference - Volume 4*, pages 20–20, Atlanta, Georgia, United States, 2000. USENIX.

[142] S. Rooney, D. Bauer, and R. Deydier. A Federated Peer-to-Peer Network Game Architecture. *Communications*, 42(5):114–122, 2005.

[143] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large Scale Peer-to-Peer Systems. In *Proceedings of 18th IFIP/ACM international conference on distributed systems platforms (Middleware)*, pages 329–350. ACM, 2001.

[144] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. *Symposium on Operating Systems Principles (SOSP)*, pages 188–201, October 2001.

[145] P. Saint-Andre. RFC 3921 - Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. Technical report, IETF XMPP working group, October 2004.

[146] A. Schmieg, M. Stieler, S. Jeckel, P. Kabus, B. Kemme, and A. Buchmann. pSense - Maintaining a Dynamic Localized Peer-to-Peer Structure for Position Based Multicast in Games. In *Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing (P2P)*, pages 247–256. IEEE Computer Society, 2008.

[147] A. F. Seay, W. J. Jerome, K. S. Lee, and R. E. Kraut. Project Massive: a Study of Online Gaming Communities. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 1421–1424. ACM Press, 2004.

[148] M. R. Shirts and V. S. Pande. Screen Savers of the World, Unite! *Science*, 290:1903–1904, 2000.

[149] R. Simon, W. Klaus, F. Marc, N. Heiko, P. Leo, and C. Georg. Peer-to-Peer-Based Infrastructure Support for Massively Multiplayer Online Games. In *Proceedings of the 4th Consumer Communications and Networking Conference (CCNC)*, pages 763–767. IEEE, January 2007.

[150] I. Stoica, R. Morris, D. Karger, and F. Kaashoek. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the 1st ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 149–160. ACM, 2001.

[151] K. Storey, F. Lu, and G. Morgan. Determining Collisions between Moving Spheres for Distributed Virtual Environments. In *Proceedings of the Computer Graphics International (CGI)*, pages 140–147. IEEE Computer Society, 2004.

[152] G. Swamynathan, B. Y. Zhao, and K. C. Almeroth. Exploring the Feasibility of Proactive Reputations. *Concurrency and Computation: Practice & Experience*, 20(2):155–166, February 2008.

[153] G. Tan and J. S.A. Stochastic Analysis and Improvement of the Reliability of DHT-Based Multicast. In *Proceedings of the 26th International Conference on Computer Communications (INFOCOM)*, pages 2198–2206. IEEE, May 2007.

[154] V. Tay. Massively Multiplayer Online Game (MMOG) - a Proposed Approach for Military Application. In *Proceedings of the International Conference on Cyberworlds (CW)*, pages 396–400. IEEE Computer Society, 2005.

[155] K. Tutschku. A Measurement-based Traffic Profile of the eDonkey Filesharing Service. In *Proceedings of the 5th Passive and Active Measurement Workshop (PAM)*. Springer, 2004.

[156] D. Twilleager, J. Kesselman, A. Goldberg, D. Petersen, J. C. Soto, and C. Melissinos. Java Technologies for Games. *Computers in Entertainment (CIE)*, 2(2):18, 2004.

[157] V. Venkataraman, K. Yoshida, and P. Francis. Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast. In *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP)*, pages 2–11. IEEE, November 2006.

[158] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta. Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, February 1992.

[159] C. Webb. *Peer-to-Peer Application Development*. Hungry Minds, ISBN: 0764549049, 2002.

[160] S. D. Webb, S. Soh, and J. Trahan. Secure Referee Selection for Fair and Responsive Peer-to-Peer Gaming. In *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation (PADS)*, pages 63–71. IEEE Computer Society, 2008.

[161] D. Whitcomb. Unexpected Crashes and Downtime Hit Multiple Servers. Technical report, News of wow.com, http://www.wow.com/tag/server-crash/, May 2009.

[162] B. S. Woodcock. An analysis of MMOG Subscription Growth, Version 23.0. Technical report, www.mmogchart.com, 2008.

[163] S. Xiang-bin, W. Yue, L. Qiang, D. Ling, and L. Fang. An Interest Management Mechanism Based on N-Tree. In *Proceedings of the Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel Distributed Computing (SNPD)*, pages 917–922. IEEE Computer Society, 2008.

[164] B. Yagoubi, H. T. Lilia, and H. S. Moussa. Load-Balancing in Grid Computing. *Asian Journal of Information Technology*, 5(10):1095–1103, 2006.

[165] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito. A Distributed Event Delivery Method with Load Balancing for MMORPG. In *Proceedings of the 4th SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 1–8. ACM, 2005.

[166] B. B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 49–60. IEEE, 2003.

[167] T. Yonekura, Y. Kawano, and D. Hanawa. Peer-to-peer networked field-type virtual environment by using atoz. In *Proceedings of the 2004 International Conference on Cyberworlds (CW)*, pages 241–248. IEEE Computer Society, 2004.

[168] A. P. Yu and S. T.Vuong. MOPAR: a Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 99–104. ACM, 2005.

[169] K.-C. Yu, H.-S. Hsiao, and F.-H. Tsai. The Implementation and Evaluation of Educational Online Gaming System. In *Proceedings of the 3rd International Conference on Information Technology: Research and Education (ITRE)*, pages 338–342. IEEE, 2005.

[170] T. Yun, L. Jian-Guang, Z. Meng, Y. Shi-Qiang, and Z. Qian. Deploying P2P Networks for Large-Scale Live Video-Streaming Service. *Communications Magazine*, 45(6):100–106, June 2007.

[171] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 594–602. IEEE, 1996.

[172] M. Zghaibeh and K. G. Anagnostakis. On the Impact of P2P Incentive Mechanisms on User Behavior. In *Proceedings of ACM Joint Workshop on The Economics of Networked Systems and Incentive-Based Computing (NetEcon+IBC)*. ACM, June 2007.

[173] K. Zhang, B. Kemme, and A. Denault. Persistence in Massively Multiplayer Online Games. In *Proceedings of 7th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*. ACM, 2008.

[174] R. Zhang and Y. C. Hu. Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Networks. In *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 172–179. ACM, 2003.

[175] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical report, UC Berkeley, 2001.

[176] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In *Proceedings of the 11th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 11–20. ACM, June 2001.