2012

A feasibility study of using peer-to-peer architecture to support massively multiplayer online role playing games

Xinbo Jiang
University of Wollongong, xinbo.jiang@gmail.com

A Feasibility Study of Using Peer-to-Peer Architecture to Support Massively Multiplayer Online Role Playing Games

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

Doctor of Philosophy

from

THE UNIVERSITY OF WOLLONGONG

by

Xinbo Jiang
Bachelor of Engineering (Honours Class I)

SCHOOL OF ELECTRICAL, COMPUTER
AND TELECOMMUNICATIONS ENGINEERING
2012

Abstract

The passed decade has seen the tremendous development and growth of online games. Among those, Massively Multiplayer Online Role Playing Games (MMORPGs) are arguably the most resource-intensive, in which thousands of players may play concurrently in a common virtual world. As a result, the current pre-dominant Central Server model may require tremendous resources to support scalable MMORPGs, and cause a potential bottleneck in terms of resource provision. In this dissertation, we conduct a feasibility study of using Peer-to-peer (P2P) architecture to support MMORPGs. In a P2P system, since participating users (peers) contribute computing and network resources for the application, the application could ideally achieve an organic growth by expanding arbitrarily without the requirement of a "forklift upgrade" to the existing infrastructure, hence addressing the scalability bottleneck faced by the Central Server model.

We begin our study with a literature review on issues and techniques for supporting online games. Latency and scalability are the two primary aspects the review is based on. Network latency heavily influences the design of online games with respect to the nature of interactivities involved in the games, and hence remains a critical metric for any technique or network design aiming to support online games. To address the bottleneck faced by Central Server model with respect to resource provision, we reviewed several scalable architectural designs, ranging from Distributed Server Architecture to Peer-to-Peer Architecture. When reviewing Peer-to-Peer Architecture, we focus on the evolution from Unstructured Peer-to-Peer Design to Structured Peer-to-Peer Design, aiming to reveal the characteristics of Structured Peer-to-Peer Network and its potential for dynamic applications such as online games.

Enlightened by the literature review, we then present a distributed game design that marries MMORPG with Pastry, a structured peer-to-peer overlay. In order to closely reflect an advanced MMORPG configuration, the distributed game design considers bandwidth and latency constraints required by the games based on actual MMORPG

traffic pattern. A simulation model is then developed to evaluate the performance of Pastry, a structure Peer-to-Peer system, in supporting the distributed game design. The simulation results have shown that Pastry performs well in distributing node stress. However, there is a bottleneck in terms of upstream bandwidth usage. We demonstrate that the bottleneck can be removed by low-cost algorithms. The results also show that using Scribe multicast tree built on top of Pastry to disseminate game traffic is an effective way to save the usage of bandwidth, which is a valuable resource in a P2P system.

We further argue that while Scribe is effective in distributing bandwidth usage for the distributed game design, it is not optimal in terms of latency performance. As a result, we propose two techniques to improve the latency performance of Scribe in the context of MMORPGs. The first technique identifies that the final hop of Scribe traffic path is largely selected without any proximity consideration and incurs the longest distance travelled. To overcome this, we introduce Proximity Neighbour Selection (PNS) into the final hop for latency improvement. The second technique builds a hierarchical two-level overlay. While PNS can be applied at both levels for latency performance, the bandwidth stress required by applications can now be distributed among the nodes in the higher level overlay. Our simulation using an internet like topology has shown that both techniques have improved the latency performance significantly, and the two-level overlay has reduced the bandwidth stress on some critical nodes in the multicast tree by up to 2.7 times, comparing with what can be achieved by a standard Scribe overlay.

To enhance the playability of MMORPGs, we introduce the concept of a seamless map that aims to make the boundaries in the game world transparent to players. MMORPG usually achieves scalability by dividing a game world into multiple locales, and each locale has a dedicated locale server that simulates a part of the game world. A seamless map requires the communication between locale servers to allow the interactions between players across neighbouring locales. In a P2P system, it is critical to seek approaches to reduce the inter-locale communication cost since locale servers are assigned to peers that can be far apart from each other in terms of latency. We formulate a Locale Assignment Problem that aims to minimize the inter-locale communication cost in a P2P system, and demonstrate through an optimisation model that the problem

is NP-hard. We then propose a low-cost heuristic for the Locale Assignment Problem. The key of the heuristic is to partition the physical network into bins so that peers in a common bin are close to each other in terms of latency proximity, as well as aggregate the partitions of a game virtual map by clustering neighbouring locales in the map. We then demonstrate that by assigning clustered neighbouring locales in the virtual map into the peers in common physical network bins, we are able to achieve similar inter-locale communication cost for the Locale Assignment Problem as that can be achieved by the optimization model we formulated, while avoiding the prohibitively high computation cost required by the latter.

We finally study the reliability aspect of a P2P system in supporting MMORPGs. In a P2P system, reliability generally refers to how system performs in response to churn – the continuous process of peer arrival and departure. In the distributed game design in P2P system, since peers contribute resources, the churn of a peer may negate game quality for other peers. To further justify the distributed game design from the reliability perspective, we develop a discrete event simulation to evaluate the churn resilience of the P2P system in supporting the distributed game design. When conducting the evaluation through the simulation, we compare the churn impact on the P2P system with different settings in supporting the distributed game design. Our general findings are two folds. First, the distributed game design based on the P2P multicast traffic model performs well in response to churn, with over 90 percentile of game sessions can achieve a pre-defined quality-of-service. Secondly, a trade-off between game features with respect to how real-time stringent actions are in the game, and the churn resilience is usually required.

We believe the findings in this dissertation will provide significant insights into supporting MMORPGs with P2P systems to address the scalability bottleneck of MMORPGs.

Statement of Originality

This is to certify that the work described in this thesis is entirely my own, except where due reference is made the text.

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

Signed

Xinbo Jiang

23 March, 2012

Acknowledgements

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

Next, I would like to thank my parents for their encouragement and support through the years. It is their consistent push that has kept me going.

Last but not the least, I would like to thank my wife Yun Su for her understanding of my commitment to the project. There have been many delayed holidays, and unrealised promises. Hopefully, I can make it up to you now.

Table of Contents

List of Figures

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Appendix A

Appendix B

List of Tables

Chapter 3

Chapter 4

Chapter 5

Chapter 6

List of Abbreviations

AOI	Area of Interest
CRF	Churn Ripple Factor
CVE	Collaborative Virtual Environment
DHT	Distributed Hash Table
DVE	Distributed Virtual Environment
FPS	First Person Shooters
MCR	Multicast Reflector
MMORPGs	Massively Multiplayer Online Role Playing Games
NPC	Non-Player Characters
P2P	Peer-to-Peer
PNS	Proximity Neighbour Selection
QOS	Quality of Service
RDP	Real Time Strategy Games
RPG	Relative Distance Penalty
RTS	Role Playing Games
TS	Transit-Stub
WOW	World of Warcraft

Chapter 1. Introduction

1.1 Background

The past decade has seen the tremendous development and growth of online games. Enabled by the Internet, and through the Distributed Virtual Environment (DVE), online games allow players to interact with each other in a virtual world in real-time regardless of geographic boundaries. The recent statistics has shown that, in America alone, online game has become a multi-billion dollar industry, and boasts tens of millions of regular players [1].

This thesis focuses on a particular game genre called Massively Multiplayer Online Role Playing Games (MMORPGs). In a typical MMORPG, a player assumes the role of a fictional character and then controls the character through a computer animation called avatar. MMORPG emphasizes real-time interactions between players. For instance, players may carry out quests through accomplishing goals collaboratively, or even choose to compete with each other.

Among genres of online games, MMORPGs are arguably the most resource-intensive, in which thousands of players may play concurrently in a common virtual world. As a consequence, the current predominant Central Server model in supporting MMORPGs usually requires tremendous resources including a large allocation of bandwidth, powerful and costly server clusters, and dedicated supporting staff, in order to maintain the scalability of a game. In the context of online games, scalability refers to the number and geographic span of concurrent players that can be supported in a game without the compromise of game quality and user experiences.

Moreover, the Central Server model in supporting MMORPGs faces the particular difficulty in resource planning. As reported in [2], the population or the number of concurrent players in online games varies significantly over both the short and the long period of time. For instance, a typical MMORPG such as EVE Online [3], the

population in the game at the peak of the day can be three times more than that at the trough of the day. Over the long period, the population of most online games goes through cycles of peak and trough, during which the peak can be three or four times higher than the trough. As a result, to maintain the scalability of online games, the resources in a Central Server model has to be over provisioned to be able to handle the peak loads.

In response to the scalability issues confronted by the Central Server model, significant efforts have been devoted to the research into the on-demand infrastructure in recent years. Among those, peer-to-peer (P2P) system leverages the resources of participating users to achieve scalability. Since the resources – including CPU cycles, data storage (e.g., RAM and hard disk), and bandwidth – are contributed by participating users, a P2P system could ideally achieve organic growth by expanding arbitrarily without the requirement of a "forklift upgrade" to the existing infrastructure, for instance, the replacement of a central server with more powerful hardware.

This thesis focuses on the challenges of converging MMORPGs with peer-to-peer infrastructure to address the scalability bottlenecks faced by the Central Server model. To that end, we present a case study in this thesis to investigate the issues related to a peer-to-peer infrastructure for supporting MMORPGs. In particular, to facilitate a P2P infrastructure to support MMORPGs, this thesis focuses on the following critical issues:

- **Bandwidth Management** – in a MMORPG environment, a client computer needs to frequently receive the state updates from the server to update the status of the game, in order to facilitate the progress of the game virtual world, as well as the interactions between the players. To ensure the quality and scalability of a game, the state updates require sufficient network resource. As a result, the efficient management of the allocated bandwidth capacity for client computers is of paramount importance for a P2P system, since client computers are usually located at the edge of network with limited bandwidth capacity allocated in comparison with a server in a Central Server model. We propose in this dissertation to use a multicast tree to multicast the traffic required by the state updates. We demonstrate that through an efficient multicast scheme that is built on top of a structured P2P overlay – a class of P2P design that applies a ring

based management protocol – a P2P system can fulfil the bandwidth requirement of scalable and highly interactive MMORPGs.

- **Latency Management** – MMORPG is a real-time applicatoin, requiring quick responsiveness to user actions. In the context of MMORPG, the responsiveness refers to the time difference between user actions and the execution of those actions in the virtual world perceived by the users. As a result, a high latency may lead to sluggish responsiveness to user actions that negates game quality. In this thesis, we propose techniques and heuristics that benefit latency management in a P2P system. It should be noted that current MMORPGs usually only allow slow-paced actions due to latency constraints. We envisage that the improved latency performance in a P2P system by the techniques and heuristics proposed in this thesis may accommodate fast actions in MMORPG to enhance user experiences and expand the scope of game design.

- **Churn Phenomenon** – in a P2P system, churn refers to the continuous process of peer arrival and departure. In a P2P system that supports MMORPG, a client computer or a peer's resources may be utilised by other peers for game state management. For instance, in the multicast scheme we propose in this dissertation, a peer may become a relay node in a multicast tree to relay game traffic to downstream nodes for game state updates. The churn of the relay node will affect the downstream nodes that rely on the relay node for game traffic, and without a suitable remedy may result in reduced game quality and user experiences for those downstream nodes. In this dissertation, we evaluate the churn impact on a P2P overlay in supporting MMORPGs, and demonstrate that a trade-off is required to balance the game features, e.g., how real-time intensive the game is, and the churn impact on the P2P system.

It should be noted that *performance and reliability* are two significant aspects to establish the feasibility of P2P systems in supporting MMORPGs. In this thesis, the study of bandwidth and latency management is to address the performance aspect, and the study of churn is to address the reliability aspect from the network perspective.

The main objectives of this thesis are:

- to propose architecture and techniques to facilitate the convergence of P2P system and MMORPGs.
- to evaluate the feasibility of P2P systems in supporting MMORPGs, both from the performance and the reliability perspectives.
- to propose techniques and algorithms to improve the performance of P2P systems in supporting MMORPGs.

1.2 Thesis Outline

This thesis is outlined as follows.

In Chapter 2, we conduct an in-depth literature review on online games to establish backgrounds and motives for this thesis. In particular, we focus on latency and scalability, the two primary aspects from the network perspective, to investigate issues and techniques pertaining to online games. With respect to latency, we first identify that the latency has direct impacts on game quality from both the consistency and responsiveness perspectives. In the context of online games, the consistency generally refers to the degree of matching between different players' views of the game virtual world, and the responsiveness refers to how quickly the user actions can be reflected in the states of the game world. We then review a number of compensation techniques for network latency and reveal that those techniques mainly aim to establish a trade-off between the consistency and the responsiveness for game quality control. With respect to scalability, we review a number of architectural designs to enhance the scalability of online games, and those mainly include Distributed Server Architecture, and Peer-to-Peer Architecture. In reviewing the Peer-to-Peer Architecture, we focus on the evolution from unstructured peer-to-peer system to structured peer-to-peer system, and reveal several interesting properties of the structured peer-to-peer system that may benefit the scalability of online games.

In Chapter 3, in light of the investigation conducted in the literature review, we propose a game scenario that marries MMORPG with Pastry, a typical structured P2P system. In modelling the game scenario, we aim to reflect an authentic MMORPG configuration. To that end, we conduct a traffic analysis on World of Warcraft, arguably the most popular MMORPG on the market. We focus on the performance study of Pastry in supporting the game scenario with respect to bandwidth and latency. In recognition of the limited bandwidth resources in a P2P system, we propose a traffic model that uses Scribe, a multicast tree built on top of Pastry, to multicast game traffic for the game scenario. Through a simulation on a 5000-node Pastry overlay, we demonstrate that Pastry can fulfil the bandwidth requirement in supporting the game scenario of 5000 concurrent players, which would otherwise incur a significant bandwidth cost in a Central Server model. However, there exists a potential bottleneck at the root of the multicast tree in terms of bandwidth congestion. We propose a low-cost algorithm to remove the bottleneck. With respect to latency, we use the overlay hops or tree length as a general indication for latency performance. We demonstrate that Pastry performs well in constraining the average number of overlay hops to below five in supporting our game scenario. However, a trade-off between game features, e.g., the number of players in a game locale, and the latency performance is usually required.

In Chapter 4, we continue our effort in studying the performance of structured P2P overlay in supporting MMORPG, and take one step further by plugging in an Internet like topology model into the Pastry overlay. We argue that a standard Pastry overlay is not optimal in terms of latency performance. Through simulation with a topology model, we demonstrate that the final hop in a Pastry overlay routing scheme dominates the routing distance due to its topologically randomised nature. Motivated by the finding, we propose a couple of techniques to improve the latency performance of Scribe, the multicast tree built on top of Pastry. The key to the first technique is to apply Proximity Neighbour Selection (PNS), a light weight technique to the final hop of a routing path in Scribe. The second technique builds a two-level overlay: the higher level among nodes located close to network access points, the lower level among nodes in a local topology domain. Both techniques have used a clustering technique that group nodes in close proximity to eliminate redundant routing path. Through simulation, we demonstrate that the two techniques have improved latency performance significantly, in comparison

with a standard Pastry overlay. Those techniques are developed in the context of MMORPG. We believe that structured P2P overlay may leverage those techniques to promote game features that are both latency sensitive and bandwidth demanding.

In Chapter 5, we present the concept of a seamless map for MMORPG that aims to make the boundaries in the game world transparent to players. MMORPG usually achieves scalability by dividing a game world into multiple locales, and each locale has a dedicated locale server that simulates a part of the game world. A seamless map requires the communication between locale servers to allow the interactions between players across neighbouring locales. In this chapter, we argue that the inter-locale communication cost is not a critical issue in Central Server model since servers are usually co-located in a LAN environment. However, in a P2P system, it is critical to seek approaches to reduce the inter-locale communication cost since locale servers are assigned to peers that can be far apart from each other in terms of latency. To that end, we formulate the Locale Assignment Problem that aims to minimize the inter-locale communication cost in a P2P system, and demonstrate through an optimisation model that the problem is NP-hard. We then propose a low-cost heuristic for the Locale Assignment Problem. The key of the heuristic is to partition the physical network into bins so that peers in a common bin are close to each other in terms of latency proximity, as well as, aggregate the partitions of a game virtual map by clustering neighbouring locales in the map. We then demonstrate that by assigning clustered neighbouring locales in the virtual map into the peers in common physical network bins, we are able to achieve a similar inter-locale communication cost for the Locale Assignment Problem as that can be achieved by the optimization model we formulated, while avoiding the prohibitively high computation cost required by the latter.

In Chapter 6, we focus on the reliability aspect of a P2P system in supporting MMORPG. In a P2P system, reliability generally refers to how system performs in response to churn – the continuous process of peer arrival and departure. The distributed game design we developed and evaluated in Chapter 3 and 4 have utilised a multicast tree as the game traffic model. In that model, a participant node may become a relay node in a multicast tree to propagate game traffic for downstream nodes. As a result, the churn of the relay node may have impact on the downstream nodes, resulting in

compromised game quality for those nodes. To further justify the distributed game design from reliability perspective, we develop a discrete event simulation in this chapter to evaluate the churn resilience of the P2P system in supporting the game design. When conducting the evaluation through simulation, we compare the churn resilience of the P2P system with different settings in supporting the game design. Our general findings are two folds. First, the distributed game design based on the P2P multicast traffic model performs well in response to churn, with over 90 percentile of game sessions can achieve a pre-defined quality-of-service. Secondly, a trade-off between game features with respect to how real-time stringent actions are in the game, and the churn resilience is usually required.

1.3 Thesis Contribution

The major contributions of this thesis are listed as follows.

In Chapter 3, this dissertation has:

- developed a distributed game design that models an authentic MMORPG configuration.
- proposed a multicast traffic model using structured P2P system in supporting the game design to achieve scalability.
- evaluated the performance of the structured P2P system in supporting the game design with respects to bandwidth and latency.
- developed a low-cost bottleneck-removal technique to remove potential bandwidth congestion at certain nodes in the multicast tree that manages the traffic for the game design.

In Chapter 4, this dissertation has proposed a couple of techniques to improve the latency performance of structured P2P system. The first technique applies a Proximity Neighbour Selection (PNS), a light weight technique, to the final hop of a routing path in the P2P routing scheme. The second technique builds a two-level hierarchical overlay for both latency and bandwidth management concern.

In Chapter 5, this dissertation has:

- introduced the concept of a seamless map for MMORPG.
- formulated the Locale Assignment Problem that aims to minimize the inter-server communication cost for a seamless map, and proved the NP-hardness of the problem.
- developed a heuristic for the Locale Assignment Problem, and through simulation, demonstrated that the heuristics can achieve the same result for reducing the inter-server communication cost as that can be achieved by an optimisation model, while avoiding the prohibitively high computation cost by the latter.

In Chapter 6, this dissertation has:

- developed a discrete event simulation model to study the churn impact on a P2P system in supporting the distributed game design for MMORPGs.
- demonstrated that the distributed game design performs well in response to churn in that more than 90 percentile of game sessions can achieve a pre-defined quality-of-service.
- demonstrated that a trade-off between game features with respect to how real-time stringent actions are in the game, and the churn resilience is usually required .

1.4 Publications Based on Thesis

X. Jiang, F. Safaei, P. Boustead, “Latency and Scalability: A Survey of Issues and Techniques for Supporting Networked Games”, 7th IEEE International Conference on Networks, ICON 2005, Kuala Lumpur, Malaysia.

X. Jiang, F. Safaei, P. Boustead, “A Performance Evaluation of Structured Peer-to-Peer Network in Supporting Massively Multiplayer Online Role Playing Games”, International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS 2006, Calgary, Canada.

X. Jiang, F. Safaei, P. Boustead, "An Approach to Achieve Scalability through Peer-to-Peer network for Massively Multiplayer Role Playing Games," *ELSEVIER Journal on Computer Communications*, vol. 30, pp. 3075-3084, 2007.

X. Jiang, F. Safaei, P. Boustead, “Enhancing the multicast performance of structured P2P overlay in supporting Massively Multiplayer Online Games”, IEEE International Conference on Networks, ICON 2007, Adelaide, Australia.

X. Jiang, F. Safaei, “Supporting a seamless map in peer-to-peer system for Massively Multiplayer Online Role Playing Games”, 33rd IEEE Conference on Local Computer Networks, LCN 2008, Montreal, Canada.

X. Jiang, F. Safaei, P. Boustead, “Churn Performance Study of Structured Peer-to-Peer Overlay in Supporting MMORPGs”, IEEE International Conference on Networks, ICON 2011, Singapore.

Chapter 2. Literature Review

The objective of this thesis is to facilitate the support of online games, in particular, MMORPGs, via the peer-to-peer overlay network. While peer-to-peer overlay inherently addresses the scalability bottleneck faced by predominant Client-Server model at present time, latency control remains a critical issue for the peer-to-peer overlay design in delivering quality of service. As a result, scalability and latency control for online games are two major aspects we explore in the problem space in this thesis, as well as, the focus of our literature review to establish the background and motives for this thesis.

2.1 Introduction

The growth and popularity of on-line games has been phenomenal in the past decade. On-line games such as the Lineage, the World of Warcraft and Counter-Strike have attracted millions of players on a global scope and achieved huge commercial success [4] [5] [6]. The underlying network support, especially the Internet, contributes significantly to this phenomenon. With the diffusion of the broadband infrastructure into the common households in recent years, the next killer Internet application might well be on-line entertainment applications that emphasize a high level of real-time interactivity.

Currently, there are three primary genres of massively distributed on-line games: Role Playing Games (RPG), Real Time Strategy games (RTS), and First-Person Shooters (FPS). Of relevance to this literature review is the difference in RTS, RPG and FPS with respect to the nature of interactivities. For example, fighting in a FPS game is real-time intensive and requires quick physical response from the players. On the other hand, fighting in RTS and RPG games is usually turn-based. In other words, the results of the fighting are predetermined by the game logic according to the level of strength of the avatar models or through somewhat random operations or factors. Statistics has shown

that in 2010 these three classes of online games counted for more than 60% sales revenue of all computer games sold in the USA [1].

While network support is one of the cornerstones that have enabled this generation of entertainment, network resource limitations impose certain constraints on on-line game design and deployment. Among these, latency is a particularly sensitive issue that, if not handled properly, will lead to poor game quality, sluggish responsiveness and inconsistency. As a result, on-line game design is heavily influenced by network latency. Furthermore, resource limitations, especially network bandwidth, computational power and latency, impose constraints on on-line game scalability with respects to the number and geographical spread of players that can be supported. As a consequence, a class of scalable designs has been proposed as augmentation and/or alternatives to current predominant Client-Server model.

The rest of the chapter is organized as follows. Section 2.2 describes the two latency-related artefacts, consistency and responsiveness, with focus on their concepts in on-line game design and influence over user experiences. Section 2.3 reviews several popular latency compensation techniques. After identifying the limitations of these compensation techniques, Section 2.4 discusses a number of scalable architectures with more details on peer-to-peer design to reveal the novelty in this new generation of concepts. We conclude in Section 2.5.

2.2 Latency, consistency and responsiveness

Online games are often based on a Collaborative Virtual Environment (CVE) [7] in that a shared virtual world is replicated at each participant's host computer to allow interactivity across geographic barriers [8, 9]. To a certain extent, the states of the replicas of the virtual world have to be consistent to allow game playability. The invocation of user actions changes the state of the world. These changes have to be reflected in other relevant players' replicas of the world as quickly as possible. However, network latency precludes instant dissemination of update information. As a consequence, a trade-off between consistency and responsiveness, the two latency-

related artefacts, is usually required. The type of trade-off that is undertaken would be application-specific.

In [10], consistency is classified into three dimensions: presentation, physics and interaction. Presentation consistency is the degree of matching between different users' views of the virtual world. The physical consistency is concerned with the laws of physics in the virtual world and whether their influence is consistent with our expectations. Interaction consistency is the degree of matching between user action and virtual object reaction. Here, the concept of consistency is defined on the basis of participant's perception of the virtual world both in space and time dimensions.

The presentation and interaction consistency are directly influenced by network latency. This is demonstrated through a shooting action example [11]: In a FPS game, player A with 100ms latency is aiming at player B when player B is running at 500 units per second perpendicular to the player A. If an instant hit weapon is used and latency is not dealt with at the game logic, player A would have to aim 50 units ahead of the player B to be able to score a hit. This is because by the time the shooting message arrives at player B's computer, the player B's spatial location would have been updated as shown in **Figure 2.1**.

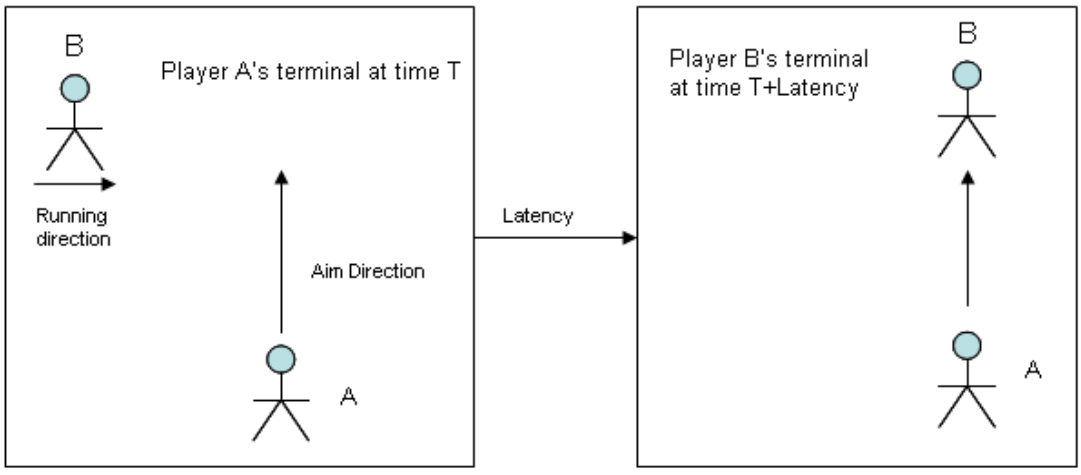


Figure2.1 Aim ahead to hit

In this case, neither the presentation nor the interaction consistency is completely preserved - the shooting action occurs at different time instants on the two terminals (presentation inconsistency), and the player A has to aim ahead at B to score a hit that would be agreed by B (interaction inconsistency). Mauve M., et al. [12] closely associate the definition of consistency with the arrival time of an operation command issued on one client to other client sites: If a user initiated an operation O at time T1, which is expected to execute before T2, the replicated world at other client sites is ensured of state consistency if O is received by every other client before T2. The consistency in this sense only concerns the final state of the replicated world as the results of the operations. A more generic "short-term inconsistency" is defined in the paper as the consequence of a late arrival of an operation that leads to at least two sites having different states at a particular instant of time.

Regardless of how it is defined, consistency is obviously desirable for on-line games. If an inconsistency occurs, depending on the category of the application, the repair scheme provided by the application might produce unwanted artefacts to the users (see section 3.2). To avoid inconsistencies, it would be desirable to synchronize the game states (see section 3.1). However, this synchronization will lead to another unwanted latency-related artifact, that is, increased response time. As defined and stated in [12], response time or responsiveness is the time difference between when an operation is issued and when it is executed. In many situations, minimization of response time and avoiding inconsistencies are conflicting goals. For instance, in the example shown in Figure 1, the shooting action by player A may be delayed for execution until the action command has reached the other client sites to maintain the consistency. However, the player will now perceive an increased response time due to this synchronization which might not be satisfactory.

In [13-16], the effects of latency on user experiences are investigated over a few popular on-line games with different characteristics. The general findings are that the level of latency sensitivity of the games depends on the nature of the interaction required by the applications. Sheldon N., et al. reported in [15] that the Warcraft III, a typical RTS game can have latencies from hundreds of milliseconds to a few seconds without

significant impacts on the performance for a player (whether winning or losing). Nevertheless, the player might perceive some delayed executions of commands when the latency reaches around 500ms. This is primarily because the nature of RTS emphasizes strategies rather than interactive aspects. Most of the operations issued by a player take seconds or even minutes to carry out; hence typical network latencies have little to no impact on the game outcomes. In [13, 16], latency effects in the Unreal Tournament 2003, a popular FPS game, are investigated. Beigbeder T., et al. [13] reported that while latencies up to 300ms have no statistical significance to influence movement actions, the accuracy of shooting actions declines sharply beyond 100ms, dropping by 50% at 300ms. In [13], it is concluded that 100ms latency is acceptable and 200ms should be avoided based on objective statistical methodologies. However, it is concluded in [16] that 60ms latency starts to introduce disturbing experiences to users based on both objective and subjective statistical methodologies. FPS games have much stricter latency requirements than RTS and RPG due to the nature of interactivities involved. As a result, the number and geographical spread of players in a typical FPS game session is usually limited.

Latency can be either compensated by smart application logic, and/or controlled through architectural design. These will be discussed through the rest of this Chapter.

2.3 Latency Compensation Techniques

Latency compensation techniques for on-line games can be generally classified into two categories. The first is *synchronization* technique (also referred to as conservative algorithm [10, 17]), and the second is *optimistic* technique.

2.3.1 Synchronization Technique

Lockstep [18] [19] is probably the simplest synchronization technique. In this scheme, if the player of the fastest host invokes an action that could change the state of the world, this change will not take effect until the slowest host has acknowledged this action. In [20], a Synchronized Messaging Service is proposed for on-line games that requires both state update fairness and player action fairness. An application independent service

module is abstracted as a network support between the server and the clients to handle message delivery service. For instance, if an update message is sent from the server to clients, the service module at the server side chooses a delivery schedule that ensures all the clients receive the message at the same time. The obvious drawback of this kind of synchronization is the impact of the slowest link and host on the performance of a whole application. However, for a certain category of games such as chess, card and puzzle games, there is a strict requirement for the ordering of events and hence absolute synchronization is needed to ensure fairness. Lockstep is also referred to as Conservative Synchronisation [19].

As a compromise between delay and response time, a “bucket” synchronization method is proposed in [21] that delays the execution of simulation for a certain amount of time (e.g.100ms) in order to bring the response time and consistency to some balance. If any inconsistency happens (a lost event or a late event arrived), it is simply ignored and game progresses as if it has never happened. In comparison with the lockstep, this bucket method can be viewed as a partial synchronization scheme that increases response time at the cost of data consistency.

2.3.2 Optimistic Technique

Optimistic technique intends to execute game optimistically and solve inconsistencies when they arise. In games that demand high level of interactivity and quick responsiveness such as FPS, the iteration of packaging and computation of updating commands (client side game logic) is executed on a per frame basis [11, 22]. For instance, a Counter-Strike game client needs to render 50-70 frames per second to achieve a state-of-the-art graphical presentation. This frequency defines the duration for client frame or iteration, resulting in 14 to 20ms frames. Ideally, the latency from the client to the server should be less than or equal to 20ms. In reality, this latency is hard to achieve for clients connecting to the server through the Internet. As a result, especially when the number of clients increases, the server might take more than a few client frames to respond with command updates. Dead reckoning [23] [11] is a typical optimistic technique that aims to use game logic at the client side to predict the states of the game to create the perception that the game is smoothly progressing regardless of

the underlying network latency. When the update message arrives, the local game state will have to make adjustments if the difference between the predicted states and the real ones is beyond a threshold. This technique compensates (masks) the latency at the cost of possible data consistency to achieve a high level of visual smoothness (high frame rate), as perceived by players.

While dead reckoning is widely used in many game designs, such as the Half-Life, QuakeWorld and Quake3, it can lead to unacceptable paradoxes. The reason is simple, if the client's prediction of a state is beyond a threshold, the state might not be fixable by the later updates. Two examples are given in [24], one is that a dead man still keeps shooting because the update message about his death is delayed due to network latency. The other is that a tank goes over a mine without being exploded because the update message moves the tank to a position that has bypassed the landmine. In [11, 25], TimeWarp is proposed to solve the possible state inconsistency as described above. It works by taking the snapshots of the past states. Whenever the state of an entity changes because of an unpredicted event, this event is also recorded. If a paradox due to data inconsistency occurs, the state of the world rolls back to the previous one (the current one is discarded) and then the current one is recomputed based on the actual events since then. The obvious drawback of this scheme is the extra memory needed to store the previous copies of the world and computational power required to do the copying and processing. Further improvement proposals are made in [24, 26], where either only a periodical snapshots are taken or an "event horizon" is defined to restrict the amount of executions that can be done optimistically.

To conclude this section, it is noted that optimistic techniques intend to establish a trade-off between consistency and responsiveness through smart logic at the application level. It should be made clear that these compensation techniques have their limitations. The reasons are two folds. First, if latency goes beyond a certain upper bound that is application specific, it is likely that none of the compensation techniques could produce satisfactory results. This can be evidenced by the experiment [27] conducted on Quake 3 (a popular FPS) servers, which reported that users prefer servers less than 150-180 milliseconds away. Secondly, as reported in [8], real-time flows such as voice, video, gestures that might be needed for on-line game design to enhance users' immersive

experiences, have a strict requirement for latency. In this case, it is less likely to achieve good results through compensation techniques. As a result, latency control remains an important issue to be addressed in a class of scalable architecture designs as discussed in the following sections.

2.4 SCALABLE ARCHITECTURES

At present time, Central Server model is the predominant model for commercial on-line games. In this architecture, a central server has the authority to execute the majority of the game logic and then send a list of objects to the client for rendering. The primary task of the client is to collect user commands and package them into data packets to send to the server for processing. The advantages of the Central Server model are well known and include simpler inconsistency management and billing model as well as simpler protection against fraud. However, besides being subject to a single point of failure, the Central Server architecture may have scalability issues.

Second Life[1] is arguably the largest Collaborative Virtual Environment at present time. With 20 million registered users, Second Life central server farm can support 50,000 concurrent users. However, this is achieved at a significant cost. Varvello M. et al [28] report that at peak time when 47,000 concurrent players are online, the aggregated traffic generated by Second Life server farm is estimated at around 13Gbps, a network cost that can only be afforded by huge enterprises. Furthermore, each region in Second Life is simulated by a server called simulator that can support a maximum number of 100 players. As a result, thousands of servers are required to support a population of 50,000 concurrent players, another significant cost that precludes small game providers from entering into the market.

In [22, 29], CPU resources are identified as the main bottleneck for game servers. Moreover, a Quake server is reported [30] that the network resource usage grows faster than linearly with the increase of the number of connected users , and possibly suffers

[1] http://secondlife.com

bandwidth bottleneck even though each user has a limited bandwidth requirement. In [8], it has been argued that latency control is of paramount importance for latency constraint applications such as on-line games, especially when real-time flows such as voice, video, gesture and haptics are required for enhanced user immersive experiences. In a Client Server model, a state update from a client has to go through the central server before being disseminated to other clients. In addition to redundant traffic, this does not work to the benefit of latency control if the game is deployed over the Internet.

An infrastructure in supporting online games is generally considered scalable if it can support a large number of players in a wide geographical span without performance degradation. From the network perspective, the scalability of an infrastructure is directly impacted by two network related artefacts: bandwidth and latency. For instance, for the Central Server model we discussed briefly in this section, bandwidth is a significant network cost in supporting a large number of concurrent players. Moreover, as we discussed in the previous sections, latency directly influences the game design related to game playability. As a result, latency control and bandwidth cost are the two chief aspects we inspect when we conduct studies on scalable designs in this chapter.

We discuss scalable architecture designs through a number of augmentations and/or alternatives to Central Server model in this section, with more details on peer-to-peer design to reveal the novelty in this new generation of concepts.

2.4.1 Distributed Server Architecture

It is natural to try to distribute the workload of a single server across a set of geographically dispersed servers close to the clients. In [31], it is proposed to have proxies located close to clients to function as an extension of the central server. The idea is that, instead of letting a central server be in charge of every game update, a local proxy can be given the task of handling some game state updates and traffic directing within the local area. Hence, the congestion at the server side is reduced and the latency within the local area is improved. In [32], a similar concept involving a central arbiter is proposed where the traffic between clients is done in a peer-to-peer fashion (see section 2.4.2), and the role of a central arbiter is to solve inconsistencies when they arise. Through a simplified model, it has been reported that a central arbiter scheme is a

compromise between the CS and the Peer-to-Peer model in terms of average latency and maximum duration of an inconsistency. Both local proxy and central arbiter can be regarded as a local server that takes part of the role of the central server.

Another distribution approach is closely associated with interest management. Although the virtual world in the game could be large in size, the limited sensing capability of an avatar suggests that a player usually has an "area of interest" and belongs to a locale [33]. Dividing the virtual world into "locales" and having one or more locales assigned to each of a set of servers is another way to design a distributed server model. There are two ways to do this. One is through server clustering [34] in CS model, where locales are distributed over a set of co-located servers. This improves scalability of server computation but the bandwidth cost related bottleneck at the server side remains, and all the traffic still has to go through the server without effective latency control. Another approach is to have locale servers distributed over geographical locations that are close to clients. As pointed out in [35], the major difficulty in this approach is to choose the best physical location for those locale servers in response to the players' movements in the virtual world. A correlation factor between a player's physical and virtual location is defined in [35] to investigate the performance of distributed locale servers through simulation. It's been reported that if the correlation factor is high, in other words, clients in the same virtual world hosted by a local locale server also tend to be in the same physical area, the average latency between clients and a server can be reduced dramatically comparing to the CS model.

A hardware called booster box is proposed in [36] to serve as a buffer between the distributed servers and the clients. It is argued that network support such as a booster box could potentially improve the performance of distributed locale servers if it combines the awareness of both the application and the network. For instance, the booster box knows both the physical and virtual location of an avatar, and hence network traffic regarding this avatar can be managed more efficiently.

In distributed server architecture, the proxy and locale server design achieve scalability through the partition of physical network and virtual world respectively. In comparison to CS model, while the total computing and bandwidth cost remains unchanged, the

major objective is to reduce latency from a client to a server. Booster box could be effective in that it addresses the gap between the application and the network.

2.4.2 Peer-to-Peer Architecture

Peer-to-peer architecture aims to utilize clients' computing and network resources to achieve scalability. We hereafter refer to a client computer together with its associated computing and network resources as a peer.

Unlike server farm or distributed servers that may be constrained by deployment flexibility, ideally, peer-to-peer systems could dynamically scale up and down with the number of players. Current peer-to-peer applications that have achieved wider applications are mainly for file storage and distribution such as Napster[2], Gnutella[3] and KaZaA[4]. Napster uses a central server to store references to music contents that are stored on peers. This is similar to Central Server model in online games. Gnutella 0.4 network works in a de-centralized manner, in which a flooding mechanism is used to search for contents. It was soon reported that Gnutella 0.4 generated too much traffic [37] [38] due to the uncontrolled number of messages exchanged. As a result, KaZaA introduced the concept of 'super peer', an extra level of hierarchy that facilitates the manageability of the peer-to-peer overlay network, and has better control of the signalling messages. This extra level of hierarchy seems to be particularly interesting for on-line games, and bears similarity to a class of hybrid peer-to-peer designs in a number of studies that are discussed in this section.

While lacking consistent and accurate definition in the literature, in the context of online games, a hybrid peer-to-peer network generally refers to a peer-to-peer network in which a small set of peers are given extra responsibilities than other peers. Those peers are usually allocated with extra network resources including bandwidth and low average latencies to other peers, and/or have extra computing resources such as CPU power. In addition, a hybrid peer-to-peer network generally does not exclude a centralized service that is not a significant barrier to scalability, e.g., a login server that authenticates peers

[2] www.napster.com
[3] www.gnutella.com
[4] www.kazaa.com

before they are allowed to enter the network. We believe that a hierarchical management is significant for online games from both feasibility and manageability perspective. We will address the reasons in the relevant context in this section. As a result, we choose to concentrate on the studies of hybrid peer-to-peer network. Hereafter, we refer to hybrid peer-to-peer network simply as peer-to-peer network.

Early peer-to-peer networks were loosely managed without well defined protocols. As a result, a peer usually had to be bootstrapped into the network via a central service provided by a well known host who keeps records of every peer in the network. The maintenance of a peer-to-peer network was usually through some probing either via specific probing messages or application related messages. Hence, this class of peer-to-peer networks are usually referred to as 'unstructured' in the literature.

The latter development of peer-to-peer design uses hash functions to map the participating peers and shared contents into an ID space. To address the manageability and maintenance inefficiencies of unstructured peer-to-peer network, this class of design applies a well defined protocol, emphasizing a 'structured' approach.

2.4.2.1 Unstructured Peer-to-Peer Network

Early work such as [39-41] typifies unstructured peer-to-peer network design in a hybrid approach in promoting the scalability of online games, or CVE in general.

DIVE [39] proposes to have game states managed by peers, eliminating the requirement for a central server. Furthermore, game states in DIVE are multicast from an updating peer to other peers through network level IP multicast. Finally, inconsistencies in DIVE are resolved through the use of Conservative Synchronization (see section 2.3.1).

In comparison with DIVE, works in [40] made two improvements for scalability concerns. First, the partition of the game world into zones, on which peer grouping is based. The benefit is apparent: only peers in a zone, a subset of the game world that constitutes An Area of Interest (AOI) in the game space, are required to communicate with each other, suggesting a reduced computing and network resource cost. Secondly, Ultrapeers, peers with more processing power and bandwidth, are used as providers for information regarding game zones, peers in a zone, and game sessions available,

relieving other nodes from exchanging too much information, hence saving total traffic cost.

In addition to partitioning the game world into Areas of Interest, a federated peer-to-peer design in [41] uses a Multicast-Reflector (MCR) to multicast state updates to peers grouped in a same Area of Interest. This is to address that IP multicast is generally not available in the current internet infrastructure. Members in the same Area of Interest use the corresponding MCR to send and receive messages to and from other members within the AOI. When a packet arrives at the MCR, if the packet is originated from a peer in the AOI that the MCR supports, it will be disseminated among the players in that group through the MCR. If the packet is from another AOI, the MCR will then use an algorithm to check the "level of affinity" of that AOI from which the packet is sent in determining whether to disseminate the packet. A MCR differs from a proxy or a locale server (see section 2.4.1) in that it does not store a whole or part of the copy of the game states. It is only required to ensure the efficient forwarding of packets to the subscribed clients. As a result, a peer with sufficient bandwidth might qualify to become a MCR. When moving to another AOI, the player will be unregistered from the current MCR and registered with a new MCR. It should be noted that the allocation of MCR is only concerned with the virtual AOI regardless of the physical location of the players, which faces the same difficulties as the distributed locale servers do when determining the allocation of MCRs.

In this class of design, the hybrid approach in the separation of peer responsibility is apparent. This is to recognize the heterogeneous nature of peer-to-peer network. We believe the hybrid approach is a significant aspect of peer-to-peer design for online games from the feasibility perspective for the following reasons:

- **The Manageability** - without a central service, the management of a large number of peers in a distributed environment will be difficult, perhaps not possible. For instance, consistency management in DIVE [39] uses Conservative Synchronization (Lockstep) (see section 2.3.1), which will degrade response time significantly if certain peers have slow links. On the other hand, through another tier of hierarchy, if a super peer is given the responsibility to handle inconsistency in an AOI, the consistency management is fundamentally no

different from what is available in a Central Server model but less scoped to a subset of the game world, e.g., a zone, and hence the management can be made possible.

- **The Security Concerns** - with the rise of popularity of online games, security has become a major concern for game developers. As reported in [42], to prevent players from cheating will be significantly more difficult in a peer-to-peer system than in a Central Server system. However, with the introduction of super peers and allowing them to manage the game states in a locale, a subset of the game world, the security measure will only have to be applied to a relatively small set of super peers instead of every peer, hence more likely to be feasible.

Another important aspect for peer-to-peer design in supporting online game is the game state management. It should be noted that in this class of designs, game states are managed by peers to eliminate the requirement for a central server. In the cases of [39, 40], without the multicast for game traffic, traffic between peers is conducted in a multi-point to multi-point fashion. Consequently, the total traffic cost in terms of bandwidth would grow quadratically with the growth of peers. This is clearly not scalable. In the case of the federated peer-to-peer in [41], the use of multicast through a super peer within an Area of Interest obviously saves total traffic cost among the group of peers belonging to that Area of Interest. However, the traffic is now conducted in a point-to-multipoint fashion, from the super peer to others. This traffic pattern may overload the super peer and restricts the scalability of the Area of Interest. As a result, it should be noted that the scalability achieved in this class of designs is the total number of game sessions, each restricted to a small Area of Interest. In other words, while the total number of players may be scalable, the number of players in each Area of Interest is limited by the capacity of the super peer. In this thesis, we argue that the scalability of Area of Interest is also important for game playability, and the key is to manage bandwidth more efficiently.

It should also be noted that latency control is not the focus of this class of design, probably due to the unpredictability of the rise of the intensity of interactivities featured by today's online games such as World of Warcraft.

If certain criteria for online games are relaxed, such as:

- the players are co-located in the same physical area and hence are close to each other in terms of latency, and
- the bandwidth requirement is low since the Area of Interest is small, and
- the security requirement is low,

then unstructured peer-to-peer may be effective in supporting a certain genre of games that fulfil those relaxed criteria. DonnyBrook [43] proposes to support First Person Shooting (FPS) games via unstructured peer-to-peer network. FPS has a stringent latency requirement that enforces players to be within 150ms away from each other to be able to participate. In addition, in FPS, a player shoots to score. At any point of time, a player can only aim at, in most case, one other player, and in rare cases, a few other players, suggesting a small Area of Interest. To make FPS more scalable, DonnyBrook further restricts the size of an Area of Interest to an Interest Set of five players to whom a player is paying attention. In addition, DonnyBrook has a set of super peers grouped into a 'forwarding pool' to forward traffic for a peer sending state update to receiving peers. A sending peer, forwarding pool and receiving peers form a multicast tree rooted at the sending peer. Given that all those peers are close to each other with an average latency less than 80ms in the simulation environment, the multicast tree is effective in distributing bandwidth while fulfilling the latency requirements. We argue that, those relaxed criteria may not be valid in MMORPG, in which players are widely spanned across the geographic boundaries suggesting latency control is of paramount importance to game playability. In addition, the size of an Area of Interest is also an important aspect of MMORPG emphasising social activity, and can not be restricted without degrading game playability. Finally, DonnyBrook has peers manage game states, which may be subject to security concerns as we discussed earlier in this section. We argue that MMORPG has more security requirements than FPS does. In FPS, each game session is independent and usually lasts 10 – 30 minutes. A player starts each session fresh and can not accumulate points from previous sessions. However, in MMORPG, game session is continuous, and many players play a number of hours for each session [44]. Players in MMORPG usually spend hours, days or even weeks to accumulate

points. If their status are tampered with or even lost, the consequence would be disastrous for those players.

2.4.2.2 Structured Peer-to-Peer Network

As discussed in the beginning of this section, the first generation of peer-to-peer designs such as Gnutella neither guarantees an answer to a query, nor the time it takes to find that answer. Those drawbacks inspired the second generation of structured peer-to-peer designs, whose main design objective is to guarantee an answer to a query with the minimum network hops, while retaining other inherent characteristics of a peer-to-peer network such as decentralized control, self-organization, adaptation and scalability. Typical examples of structured peer-to-peer design are Can [45], Chord [46] and Pastry [47]. While being different in design details, their common feature is that peers form an overlay network at the application level and each node stores a small amount of routing information to allow message routing.

Can maps a node into a virtual d-dimensional Cartesian coordinate space, where each node owns a distinct zone and maintains a small routing table containing the IP addresses and zone coordinates of its immediate neighbours. A data item could be hashed into a key that matches a zone coordinate and then assigned to the zone owner. **Figure 2.2** shows a 2-D space partitioned between 6 Can nodes and a sample routing path from node A to F. Individual nodes maintain O(d) neighbours and the average path length is $O(d(n^{1/d}))$.

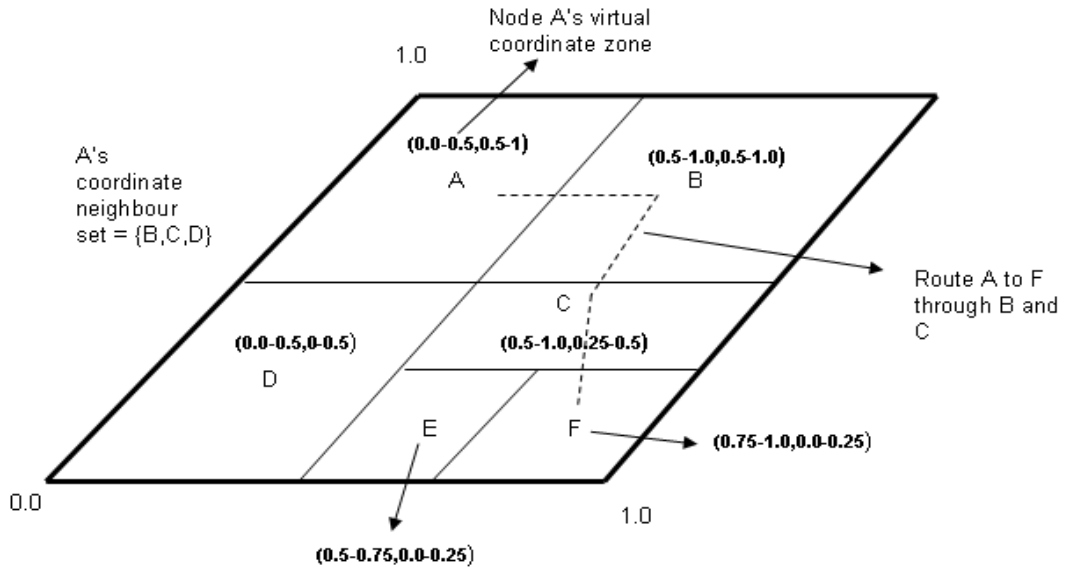


Figure2.2 Can co-ordinate based overlay with 6 Nodes [45]

Pastry's core task is to route a message with a key to a node with an ID numerically closest to that key. ID assignment for nodes in Pastry can be viewed in a ring space, whose size is determined by the number of digits assigned to that ring. A node's ID is assigned by hashing its IP address or Public Key. The hashing function (SHA-1) guarantees with high probability that the nodes will be evenly distributed around the ring. Each node in the Pastry ring space maintains a small routing table that facilitates the routing of a message. Given a message with a key, a node along the route will try to relay the message to another node whose ID shares one more prefix with the message key than itself as shown in **Figure 2.3**.

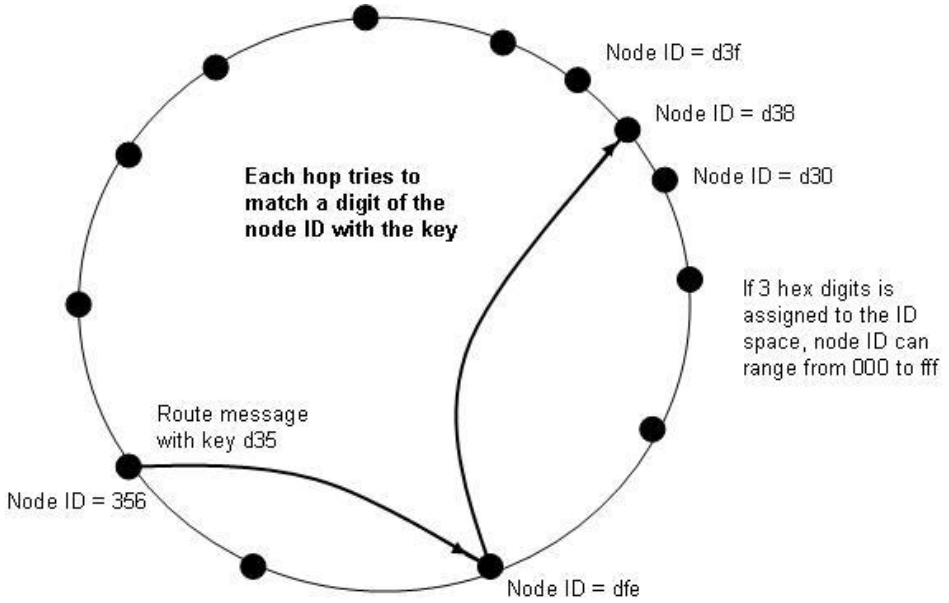


Figure 2.3 Pastry routing example

The expected routing hops for a given message is $\log_{2^b} N$ where N is the total number of nodes in the overlay network and b (usually chosen to be 4) is a configuration parameter that determines the numbering base (e.g., hexadecimal) of the routing table. Another way to look at it is that if a message with a key "d35" is to be routed and if every hop can match one more digit of that key, then the maximum hop would be the number of digits of the key. A detailed Pastry routing example is given in Section 3.2 when we explain the multicast technique that is based on the Pastry routing scheme.

The design of Chord is similar to Pastry. Instead of matching a digit per hop for a message key, Chord routes the message to the nodes whose IDs are numerically closer and closer to the key. Given that the values of IDs in this class of design are usually the results of hash functions to achieve uniform distribution of IDs in the ID space, this class of structured peer-to-peer design is also referred to as Distributed Hash Table (DHT) based peer-to-peer overlay in the literature.

In Section 4.5, Chapter 4, we further conduct a comparison between different structured peer-to-peer systems, and briefly review a number of works that aim to enhance the performance of the structured peer-to-peer systems.

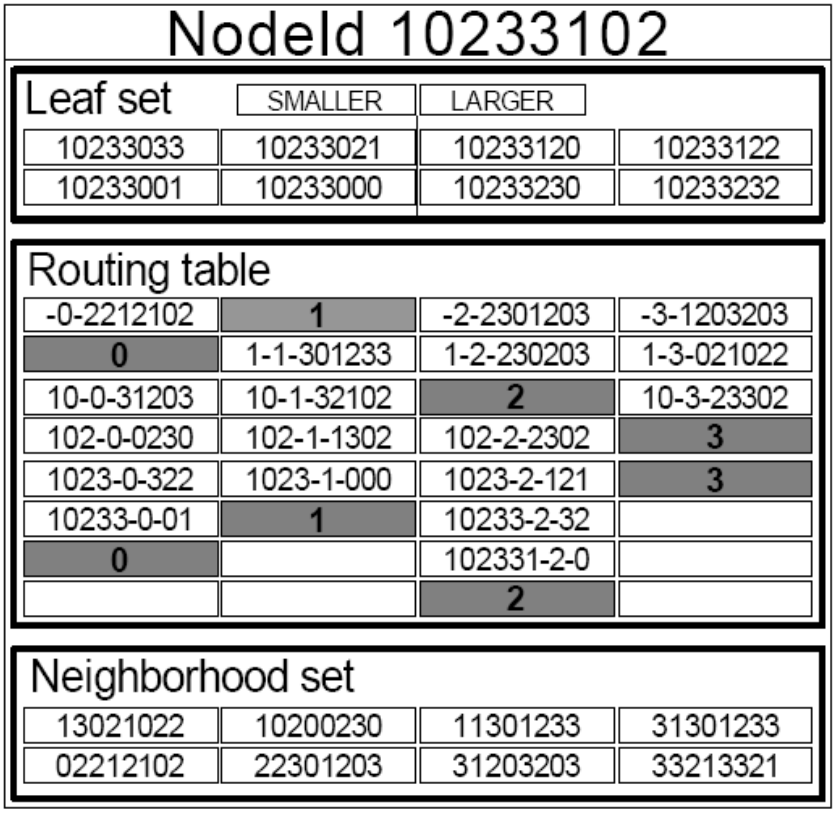
This category of peer-to-peer design is mainly concerned with load balancing through the uniform distribution of keys among peers, and quick file retrieval with minimum number of network hops. Clearly, it is not designed for latency constraint applications such as on-line games. For instance, Can, Pastry or Chord make no or little effort to achieve network locality. In other words, nodes close in the ID space could be far apart from each other geographically or in terms of network latency. Consequently, the routing algorithm might force a node to go through a neighbouring node in the ID space but too far away geographically to search for a piece of information.

However, there are a number of attractive key features in this design such as decentralized control, self-organization, adaptation and scalability. Those key features distinguish fundamentally a structured P2P overlay from a traditional central server based network, and will be discussed briefly as follows.

Scalability – the establishment of efficient communication between nodes in a structured overlay only requires a small set of entries maintained in the routing table of a node. For instance, in a Pastry overlay of 10^6 nodes, a routing table in a node contains on average 75 entries and the expected routing hop is 5, whilst with 10^9 nodes, the routing table contains on average 105 entries, and the expected routing hop is 7. A sample routing table in a Pastry node is given in **Figure 2.4**. Note that in the routing table has three sections: the first section Leafset contains nodes that are numerically closest to the sample node, the second section Routing Table contains nodes whose numerical id matches the sample node id with a certain number of digits. For instance, the nodes in the first row of the Routing Table match the sample node id with the first digit, and the nodes in the second row of the Routing Table match the sample node id with the first and the second digits. The third section Neighbourhood Set contains those nodes that are close to the sample node according to Proximity Metric.

Decentralised control, self-organisation and adaption – for instance, in Pastry, a newly arrived node A only needs to know an existing node B in the network to be

bootstrapped into the overlay network. Ideally, node B would be close to node A according to the proximity metric, e.g., latency. Node A will then ask node B to route a special join message to an existing node C whose id is numerically closest to node A. In response to receiving the "join" request, nodes B, C, and all nodes encountered on the path from B to C send their state tables to A. The new node A inspects this information, may request state from additional nodes, and then initializes its own routing table. When a node fails or departs without warning, other nodes having that node in their routing table require to replace the entry with a live node, and the procedure in fixing the routing table is done in a lazy fashion. For instance, a failed node will be detected while a message being delivered is using that node in the routing path. On detection of the node failure, an alternative path will then be chosen to deliver the message. In the mean time, nodes will launch the procedure to replace the failed node in the routing table by contacting entries the routing table located in the same row as the failed node, and with high probability, a candidate live node will be found to replace the failed node. For details, please refer to [47].



NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Figure 2.4 A sample routing table for a Pastry node [47]

To summarise, in an unstructured peer-to-peer network, a peer needs to contact a well known peer to be bootstrapped into the network. However, in a structured peer-to-peer overlay, a peer can be bootstrapped into the network by contacting any peer in the overlay, emphasizing *decentralized control and self-organization*. In addition, with the high rate of peer joining and leaving the network, the maintenance protocol only incurs a small network traffic cost to adjust the small routing table stored in each peer to maintain the stability of the overlay, benefiting *adaptation*. Furthermore, the overlay network can grow to a large number of nodes at a small cost to individual peer's resources. This is because the routing table in each node grows much slower than the number of nodes in the network, contributing to *scalability*.

In [48], a simple Role Playing Game called SimMud is simulated leveraging a Pastry network. SimMud achieves scalability by dividing the game into locales and appointing a "coordinator" for each locale conforming to a hybrid approach. It is noted that a coordinator multicasts game state update to participating peers, through a multicast tree built on top of Pastry, in contrast to the point-to-multi-point multicast of game state in [41] as we discussed in last section. This is to take advantage of Pastry having a well defined protocol that facilitates the establishment and maintaining of a multicast tree with a low cost. It is also interesting to note that the approach SimMud uses to backup the game states. In compliance with the Pastry routing algorithm, an object in the game with key K is stored in the node whose ID is numerically closest to K. This node then becomes the coordinator for this object. The next numerically closest node B will store a replica of the object. If the coordinator K leaves the network, B will automatically be promoted to the coordinator for the object. The underlying Pastry network ensures that all peers interested in that object can detect the new coordinator quickly, and hence retrieve the object states if a coordinate leaves the network. This is to leverage the de-centralized and adaptation feature of Pastry design.

Badumna [49] is a network engine that facilitates the development and deployment of MMORPG on DHT based structured peer-to-peer network. Badumna partitions game world into smaller units called cells, and assigns a peer to be the Cell Server. This is similar to Local Server concept. However, maybe due to commercial concerns, it is not revealed in the literature how a Cell Server manages game traffic, in particular,

how Badumna utilises the underlying structured peer-to-peer overlay to manage network resources such as bandwidth in response to the game traffic requirements. Moreover, Badumna recognises the potential bottleneck of a Cell Server when bandwidth requirements rise to a threshold. When this threshold is reached, Badumna proposes a Gossip Mode, in which peers participating in a cell start to communicate with each other instead of through the Cell Server. Again, the traffic pattern in Gossip Mode is not revealed. Badumna addresses a number of issues related to peer-to-peer design integration with commercial platforms. Among those issues the most critical ones are control and security. For instance, commercial game providers require complete network monitoring, transaction logging, and support for secure transaction, suggesting a peer-to-peer design for supporting online games may require a compromise between game state distribution and network resource utilisation if aiming for commercial integration.

2.5 Conclusions

In this Chapter, we have provided a literature review of mechanisms and techniques for supporting on-line games. Latency and scalability are the two primary aspects on which this literature review is based.

Latency heavily influences current on-line game design with regards to the nature of interactivities involved in the games. As a result, latency compensation techniques usually intend to establish a compromise between consistency and responsiveness, the two latency-related artefacts. Due to the limitations of these compensation techniques, latency remains an important issue to be addressed in scalable architecture designs.

To address the drawbacks of the current Central Server model in supporting online games, scalable architecture design distributes the computation over clusters of servers or peers. The scalability is usually achieved through the partition of the virtual game world or physical network. In the distributed server design, this has resulted in systems such as the locale or proxy server. In the peer-to-peer design, in addition to the partition approach, hybrid design introduces the concept of 'super node' to allow network manageability.

DHT based peer-to-peer design such as Pastry emphasizes a structured approach through the use of a well-defined protocol, contributing to a set of features such as decentralized control, self-organization, adaptation and scalability. Although it is not designed for latency constraint applications, those features are desirable for online games from the perspective of peer-to-peer support.

Chapter 3. A Performance Evaluation of Structured P2P Overlay in Supporting MMORPGs

Abstract

Scalability is a critical issue for Massively Multiplayer Online Role Playing Games (MMORPGs). To address the scalability drawbacks of current central server model, we present a distributed game design to support MMORPG through Pastry [47], a structured peer-to-peer overlay. In order to closely reflect an advanced MMORPG configuration, the distributed game design considers the bandwidth and latency constraints required by the games based on actual MMORPG traffic pattern. A simulation model is developed to evaluate the performance of Pastry in supporting the design. Results show that Pastry performs well in distributing node stress. However, there is a bottleneck in terms of upstream bandwidth usage. This bottleneck can be removed by low-cost algorithms proposed in this Chapter. Results also show that using Scribe multicast tree built on top of Pastry to disseminate game traffic is an effective way to save bandwidth usage. In addition, Scribe multicast tree scales well in supporting the distributed game design with respect to the tree length. When a tree size increases five times to support a large group size of concurrent players, the length of the tree only increases less than two levels.

3.1 INTRODUCTION

Among genres of online games, scalability is a particularly critical issue for some Massively Multiplayer Online Role Playing Games (MMORPG), in which thousands of players need to be supported concurrently in a large virtual world. In this respect, scalability refers to the number and geographic spread of players that can be supported, and is the focus of this chapter.

A typical MMORPG usually follows a storyline in a medieval fantasy world, in which a player can take adventures through the control of an avatar model – a representation of a role character in the virtual world. An attractive feature of MMORPG is the huge size of the map that constitutes the fantasy world and a large number of players, usually in the range of thousands that can potentially interact with each other. For instance, a group of people can form a team to do a quest together, or fight a monster, or even fight against other groups of players. **Figure 3.1** shows an example of an area in a virtual world, in which a number of avatars were gathering for potential game activities.

Figure 3.1 An example of a game virtual world

Currently, MMORPGs are predominantly supported by a Central Server (CS) model, in which a central server has the authority to execute the majority of the game logic and then send a list of objects to the clients for rendering (state update). The primary task of a client is to collect the user commands and send them to the server for processing. While computational power and bandwidth requirement in a CS model for a single client is minimal, when thousands of players join the game, significant resources including the computation power, bandwidth and dedicated support staff required at the server side will potentially restrict the scalability of the game.

Central Server model achieves scalability by employing server clusters. For instance, a popular locale-based approach divides a huge map into sections called locales, and employ different servers to simulate each locale and hence distribute the computational stress over a cluster of servers. However, in addition to a central point of failure, the possible bandwidth stress at the server side cannot be avoided. Furthermore, this model lacks flexibility and the resources have to be over-provisioned to handle peak loads.

There is a significant body of work related to a class of scalable architectural designs to address the scalability issues in the CS model. A detailed survey regarding these designs has been conducted in Section 2.4 in the Literature Review Chapter and references included therein. Among these designs, there is a class of peer-to-peer systems that advocate the use of client computing resources (peers) to achieve scalability. Knutsson B. et al. [50] proposed the use of Pastry, a structured peer-to-peer overlay to address a couple of critical challenges for a peer-to-peer system to support MMORPG. The first is *performance*: the game design must consider limited computing resources of peers, especially bandwidth, since peers are usually located at the edge of the network. The second is *availability*: a game state replication algorithm is required to tolerate possible high failure rate of peers (e.g., peers go offline). With respect to the performance issue, the paper based its study on SimMud, a "purposely-simple" game implemented for research purposes. However, the latest development of MMORPGs such as World of WarCraft usually have complicated features in terms of player's choice of actions, and are enriched with state-of-the-art 3D graphics. As a result, the demands for bandwidth of peers will directly influence the game playability. To further justify a Pastry overlay

in supporting MMORPG, there is a clear need to consider actual traffic patterns of these MMORPGs and their impacts on peers such as bandwidth stress.

In this chapter, we develop a peer-to-peer support for a distributed game design based on Pastry that considers actual MMORPG traffic pattern. In addition to investigate the bandwidth distribution among peers when they are mapped onto a Pastry overlay, we further envisage a map configuration as part of the design in considering latency constraints, which are critical in determining the nature of interactivities that can be supported in a game. We develop a simulation model to evaluate the game design quantitatively. Our intention is to evaluate the performance of Pastry, a typical structured Peer-to-peer overlay, in supporting the distributed game design that truly reflects a typical MMORPG configuration. We focus on the *performance* aspect of peer-to-peer support for MMORPGs in this chapter and address the *availability* aspect in Chapter 6.

The rest of the chapter is organized as follows. In Section 3.2, we give a brief introduction to the Scribe Multicast Tree established upon Pastry. In Section 3.3, based on the traffic pattern and design features of MMORPG, we present a distributed game design that is established upon Pastry to support MMORPG. In Section 3.4, we present the simulation results that evaluate the Pastry network performance in supporting the distributed game design. We finally conclude this chapter in Section 3.5.

3.2 SCRIBE MULTICAST TREE

Given the current MMORPG traffic pattern and general asymmetric allocation of upstream and downstream bandwidth for a peer, it would be desirable to use a multicast tree to disseminate state update messages from a locale server to peers who play in the locale in order to save bandwidth of peers. It should be noted that, in a peer-to-peer environment, the duty of a locale server is assigned to a peer in the overlay. In Section 3.3.1, we discuss in details the rationale behind the use of a multicast tree as part of our proposed game design. Here, we first sketch Scribe [51] , an application level multicast tree built on top of Pastry.

We have introduced Pastry as a typical structured peer-to-peer overlay design in Section 2.4.2.2 in Literature Review chapter. The establishment of a multicast tree on top of Pastry is a straightforward process. If a node is selected as the root of a multicast tree, the nodes that are members of the tree find their paths to the root of the tree through the underlying Pastry prefix match routing approach as explained in Section 2.4.2.2. If two paths overlap at a node, the multicast tree is then established through reverse path forwarding scheme and the overlapped node belonging to both paths becomes the interior node of the tree. An example is given in **Figure 3.2**. Both nodes 356 and 467 are members of the tree rooted at node D38. The path found by node 356 through the underlying Pastry prefix match approach is 356 -> DFE -> D38. The path found by 467 is 467->DFE->D38. Since both paths overlap at the node DFE, the node DFE becomes the interior node of the multicast tree.

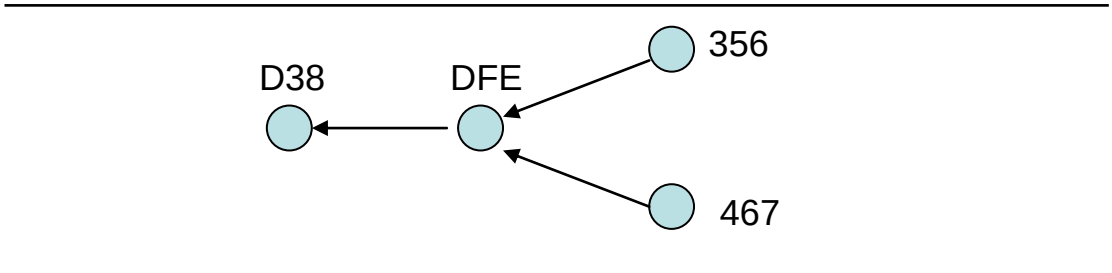


Figure 3.2 A Scribe example

Simulation results in Section 3.4 in this chapter will reveal that, due to the random nature of Pastry overlay in choosing a path between two peers, the Scribe multicast tree

is not optimal in terms of bandwidth distribution given the MMORPG scenario. Some peers, especially the roots of trees are likely to be subject to high upstream bandwidth stress. However, low-cost algorithms can be used to remove this bottleneck. This, in turn, takes advantage of the random nature of a Pastry network.

3.3 Peer-to-Peer SUPPORT for MMORPG

In this section, we present a distributed game design to support MMORPG through a Pastry overlay. Our main objective is to take advantage of Pastry, a structured peer-to-peer overlay, by distributing computation and bandwidth stress over peers. In our design, we consider the game bandwidth usage based on actual MMORPG traffic pattern and latency constraints of disseminating state update information. Our intention is to have the design closely reflect a typical MMORPG configuration.

3.3.1 Traffic Pattern of MMORPG

3.3.1.1 Traffic Pattern of Lineage II

Kim J., et al. [52] conducted a traffic analysis on Lineage II, a leading 3D MMORPG developed by NCsoft, which has reported two million registered users and 150,000 concurrent players. We summarize the traffic analysis that is of relevance to our design as follows.

Firstly, there is a substantial asymmetry between the upstream and downstream bandwidth stress for the game traffic. For a single client-server connection, the average traffic generated by a client is 2kbps, while the average server generated traffic is 20kbps. This 10 times asymmetry is associated with Central Server model. In a typical client/server model, the client computer simply samples user inputs to send to the server for simulation. This results in small sized client packets and minimal bandwidth requirement. However, on the server side, a list of objects representing the new state of the virtual world needs to be sent to the client resulting in a much larger bandwidth requirement. As reported in [52], when the maximum load of 5000 players is reached at a central server hosting Lineage II, the upstream bandwidth stress of the server is 140Mbps, while the downstream bandwidth is a mere 9Mbps. It should be noted that

140Mbps is clearly the capacity of a T4 backbone link out of the central server that, given this particular MMORPG traffic requirements, limits the maximum game scalability to 5000 direct connections.

Secondly, the relationship between the number of users and bandwidth usage at the server side is linear. In other words, while the number of player increases from 2000 to 5000, the average per connection upstream and downstream bandwidth stress at the server side for each client/server pair remains at 20kbps and 2kbps respectively. This suggests that the technique of interest management is used in the Lineage II design whereby a player only receives a subset of update information of relevance to him in his *area of interest (AOI)* [53]. Hence, even with the increased number of players in the world, the central server upstream bandwidth for an individual connection can be maintained within a certain range.

It should be noted that the household Broadband connections are generally asymmetric – the download capacilty outperforms the upload capabcity significantly. As game scale increases, the aggregated upload capacity or upstream bandwidth stress increases linearly. As a result, for peer-managed games, the key limitation is upload capacity or upstream bandwidth for peers [43]. Hereafter, we use upstream bandwidth and upload capacity interchangeably to refer to the data upload transfer speed in terms of bit rate, e.g., kilo-bits per second (kbps).

In light of the above analysis, because of the substantial asymmetry between the upstream and downstream bandwidth stress for the game traffic, it would be desirable to have a multicast tree to disseminate state update information from locale servers to peers in order to save bandwidth usage. Considering a practical peer with popular ADSL setup of a 1000/256kbps link (1000kbps downstream bandwidth or download capacity, 256kbps upstream bandwidth or upload capacity), the downstream bandwidth capacity can allow 500 connections if each requires 2kbps for sending action command as in Lineage II. However, the upstream bandwidth 256kbps can only allow 12 connections if each requires 20kbps for sending state update information. On the other hand, the technique of interest management has made it feasible to organize players with common AOI into groups. This self-organizing property of MMORPG can obviously take advantage of the self-organizing characteristics of a Pastry peer-to-peer

overlay. In particular, peers with common AOI can form peer groups to receive common state update information. In our simulation model, peers in the same locale are organized into a common multicast tree to receive state update information from a common locale server. It should be noted that in an advanced MMORPG such as 3D Lineage II, the size of a locale could be quite large and may constitute many AOIs. The level of granularity with respects to the partition of a huge fantasy world into locales and further into AOIs is obviously a design issue that must consider a server's computational power and bandwidth capacity.

It should be noted that in an advanced MMORPG such as World of WarCraft, the size of a locale could be large and may constitute many AOIs. A locale server in a central server cluster computes an AOI for a player dynamically as he moves around the game world in order to control the bandwidth usage of state update flow for that player. For the distributed game design using Scribe, the partition of a locale to AOIs is omitted for two reasons. First, the dynamic computation of an AOI for each player as he moves around is expensive. Second, when locales are distributed among peers, the bandwidth usage for a locale is limited to the state update requirements for that particular locale, whereas in a Central Server model, multiple locales share the same chunk of bandwidth available. Consequently, when locales are distributed among peers, the cost of partition of a locale into AOIs might outweigh the bandwidth that could be further saved by the partition. As a result, for the rest of the chapter, we refer to a locale as an AOI so that each peer in the Scribe multicast tree receives the same copy of state update messages from a locale server without the need for further partition.

Our simulation (see Section 3.4) creates a Pastry overlay of 5000 nodes to match the maximum capacity of Lineage II server and assign 127 peers as locale servers. Each peer is assumed to have a popular DSL link of 1000/256kbps. Because of the traffic asymmetry discussed above, we use Scribe to multicast state update information to peers. Figure 3.3 demonstrates the traffic flow in our game design.

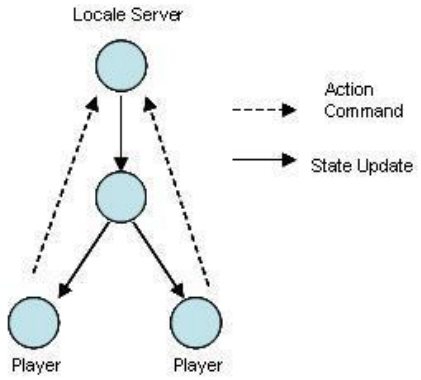


Figure 3.3 Game traffic flow

As shown in **Figure 3.3**, if a multicast tree is used to disseminate state update information and the two players have a common AOI and hence are connected to the common locale server, the locale server only needs to send one copy of the update message instead of two as in the Central Server model. If a large number of players are connected to the locale server as shown in **Figure 3.3**, the multicast technique could potentially save a significant amout of upstream bandwidth required by the locale server to disseminate the game state update for players in the locale.

3.3.1.2 A Traffic Trace at the Client Side

In [52], it is not clear if the bandwidth of a connection for a state update varies as the environment of the virtual world changes. An educated guess would be that the bandwidth requirement is closely associated with the nature of a locale, that is, the number of objects in that locale and the state update rate for those objects. In a typical MMORPG, the objects in a locale include system-predefined objects such as non-player characters (NPC) and avatars controlled by real players. With respect to the state update bandwidth requirement, while the former can be controlled through system design and hence remain relatively static, the latter is more dynamic because of the unpredictable behaviors of players. For instance, the number of players in a locale at a time cannot be predicted by the system. Hence, we believe that state update bandwidth requirement is more sensitive to the number of players in a locale. To investigate this, we conducted a traffic trace at the client side for World of WarCraft (WOW), which is arguably the most popular MMORPG on the market at present.

While the world in WOW is huge for a player to explore, it is constituted of small towns, cities and wilderness, which has made it possible to apply locale-based approach in the game design. We first conducted a real-time traffic trace by taking an avatar from place to place, namely Goldshire, Stormwind and Ironforge. **Figure 3.4** shows the traffic trace.

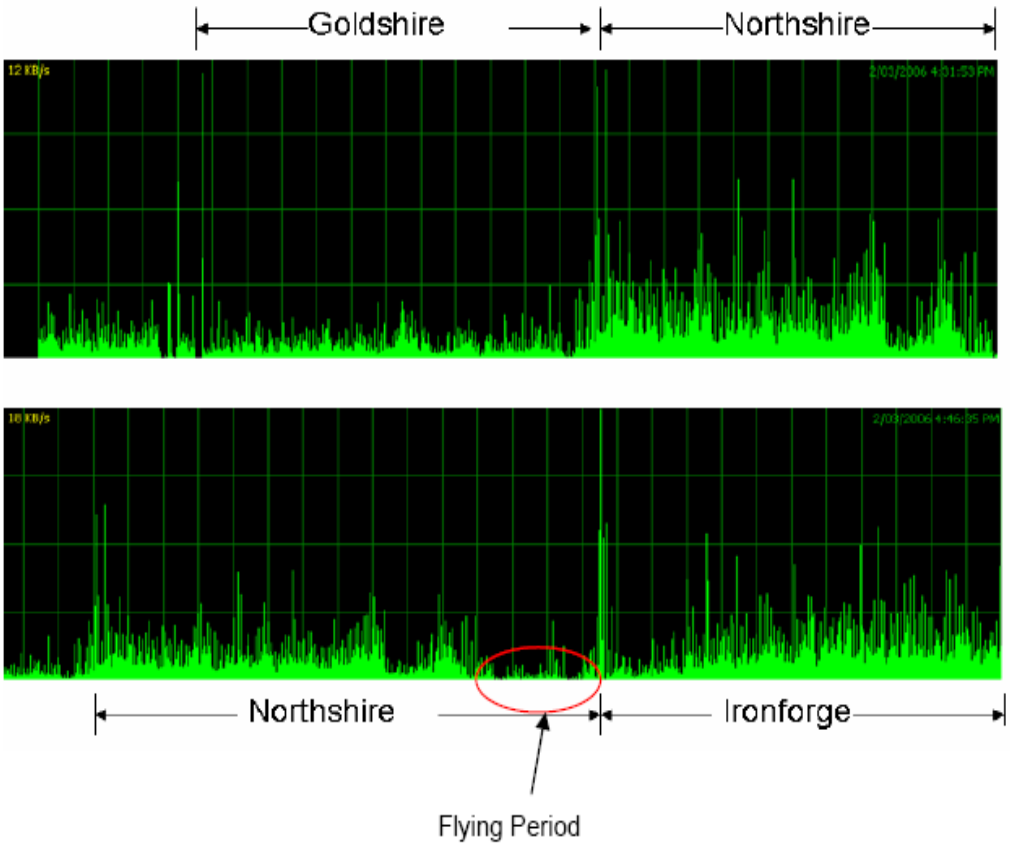


Figure 3.4 Traffic trace

In **Figure 3.4**, it is shown that each section of the trace has clear boundaries distinguished by the bandwidth surges at the transition, which occur when the avatar moves from one place to another. For instance, when the avatar moves from Stormwind to Ironforge, he may take a ride on a bird through a wide area of wildness. The period of flying is indicated by the red circle in the figure, which clearly shows a drop of bandwidth usage. Before entering Ironforge, there is a peak of bandwidth usage for a short period of time lasted about 20 seconds. This peak, we believe, is the downloading and/or initiating of locale-specific objects and their status. It should be noted that the section for Stormwind is the same trace in both the top and bottom charts in Figure 3.4

presented with different scales to show the transition from one place to another. This transition peak gives potential evidence of a locale-based design approach might habe been used in WOW.

To investigate how sensitive the bandwidth usage is to the number of players in a locale, we focused on a particular sector of Ironforge called the Commons. The Commons is a popular place where players can do their banking, auction items or simply convene for social matters. In the Commons, players come and go with a noticeable population change throughout the day. By positioning our controlled avatar in the square, it is possible to observe other avatars to the far side moving away from the Commons and disappearing into other places in Ironforge. Hence, it is a good representation of an AOI that can be implemented in MMORPG. We take a window size of five minutes to conduct the sampling. All the sampling windows are relatively evenly distributed over the daytime. Each sampling point is an average bandwidth usage versus average number of players counted in the Commons during the window duration. There are altogether twelve sample points taken and the results are plotted in **Figure 3.5**, which shows a linear pattern between the number of players and the bandwidth usage and the level of linearity being achieved is 0.83.

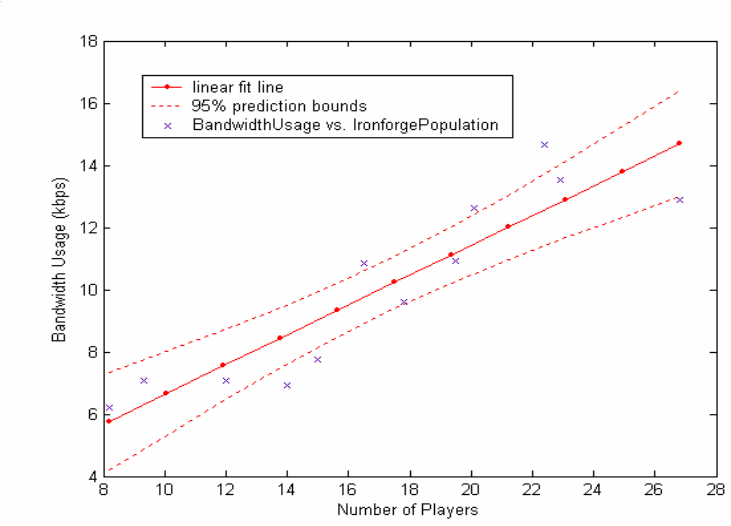


Figure 3.5 Bandwidth vs. number of players

In light of the above trace, to simplify the simulation, we envisage that the bandwidth requirement for state update correlates with the number of players in the locale linearly.

Hence, we categorize the 127 locales into four types of maps, where each map has a state update bandwidth requirement linearly dependent on the number of players that can be supported. For instance, in **Figure 3.5**, when 20 players are in the Commons, bandwidth usage reaches approximately 10kbps if a linear pattern is assumed. We then assign 10kpbs as bandwidth requirement to Map Type 1 to be able to support a maximum of 20 players. Map Type 2 needs to support a maximum of 40 players and the bandwidth requirement increases to 20kbps. For details, please refer to **Table 3.1**. Note that, the bandwidth requirement in kilo-bits per second (kbps) is projected from the traffic trace (**Figure 3.5**) we conducted in World of WarCraft to mimic a real traffic in a popular MMORPG.

It should be noted that while the traffic trace in **Figure 3.4** has given evidence that a locale based approach might have been used for WOW, it is not the direct reason we have opted a locale approach for the distributed design. A locale based approach is natural for a game to achieve scalability, for details please refer to our discussion in Literature Review, Section 2.4.1.

It should also be recognised that our traffic trace in **Figure 3.5** is brief in that we only conducted the traffic trace in a part of the game world. In a comprehensive traffic trace conducted in Lineage II [52], a game similar to World of WarCraft, a similar linear pattern between the bandwidth usage and the number of players in the game world can also be found. However, it can not be ruled out that the traffic might show different pattern in the other part of the game world in World of WarCraft, given that user behaviour and game nature differ between games. It will remain our future work to further investigate the traffic pattern of World of WarCraft in different part of the game world. While the traffic trace gives us reasoning for the distributed design, it is not the focus of this chapter.

It should be noted that the state update bandwidth requirement in **Table 3.1** is projected from the traffic trace in **Figure 3.5**. This allows us to investigate whether or not a multicast tree built on top of a P2P system can manage the bandwidth requirement effectively. It should be recognised that the justification of using a multicast tree is not solely dependent on the traffic pattern in **Figure 3.5**. The efficient management of bandwidth in a P2P system is fundamentally critical in supporting MMORPG. The

pattern in **Figure 3.5** gives us reasoning to apply the technique of multicasting. We believe the projected traffic pattern in our distributed game design, based on **Figure 3.5**, is conservative in testing the targeted performance of the multicast tree, and may provide us with a baseline to experiment with the P2P system for performance concerns, e.g., whether or not a multicast tree built on top of P2P system can support the potential significant bandwidth requirement.

Table 3.1 Distributed locale setup

	Map type 1	Map type 2	Map type 3	Map type 4
State update bandwidth requirement (kbps)	10	20	30	40
Interactivity	Player vs. Player fighting	Player vs. Non-player fighting	New player training school	Market city
Real-time sensitivity	High	Medium	Medium	Low
Number of maps (locales)	60	35	20	12
Number of players supported (group size)	20	40	60	100
Total number of players (group size)	1200	1400	1200	1200

3.3.2 Latency consideration

We also consider the real-time constraints of interactivities in a map, which is closely related to network latency. While certain online games such as the Counter Strike require latency of less than 150ms to achieve good playability [54], MMORPG can usually tolerate latencies up to several hundred milliseconds without severe impact on the quality of game. In the case of Lineage II, the probability distribution of game traffic latency (RTT) is concentrated at 200 milliseconds. This is because the current MMORPG design does not require low latency. For instance, the results of a fighting in MMORPG are usually predetermined by the game logic according to the level of strength of the avatar models or through somewhat random operations or factors. Hence, fighting in MMORPG does not require strict real-time interaction.

For the distributed locales in a Pastry network as we proposed above, a locale that supports a small number of players are more suitable for interactivities with stringent real-time constraints. This is because when using Scribe Multicast Tree to disseminate state update information, a tree established by a small number of players can be guaranteed a small maximum length. For an overlay size of 5000 in our simulation, maximum length for a tree without using a bottleneck removal algorithm is less than four. In other words, in a small sized Scribe Multicast Tree, a peer can reach its locale server (the root of the tree) in less than four hops. This is clearly to the advantage of latency control if latency for each hop can be controlled within a certain range. Although a Pastry overlay can not guarantee an upper bound for maximum latency between a pair of peers due to its random nature and lack of network topology awareness, we believe techniques can be developed to control the Pastry network latency. For instance, a binning scheme [55] uses landmarks to localize nodes in a structured peer-to-peer network, and hence improves the network topology awareness and benefits latency control. In our simulation, we assume that less hops generally correspond to less latency.

Based on this assumption and our gaming experiences, we then envisage that locales with small capacity can support interactivities that have real-time constraints such as player versus player fighting (Map Type 1), and locales with large capacity will only

support interactivities of less real-time nature, such as players trading goods in a market (Map Type 4). This is because, as further revealed in the simulation results in Section 3.4, the length of trees supporting a large number of children would be longer. From a child's perspective in the multicast tree, this will result in extra hops to the root of the tree, and hence possible extra latency.

Table 3.1 gives the details of the distributed locale setup based on discussions presented in this section.

3.4 Simulation Setup and Results

3.4.1 Simulation Setup

We have developed a model that simulates a Pastry network of 5000 nodes to match the maximum capacity of a Lineage II central server. We randomly assign a latency value in the range of 3-100 milliseconds between each pair of nodes as in [50]. A node joining the overlay to establish its leaf set, neighbour set and routing table. The establishment of the leaf set and neighbour set for each node follows the protocol and procedures as described in Pastry (see **Section 2.4.2.2**). However, in establishing the routing table for a node, given the relatively small size of our simulated overlay, we envisage that a central service can be made available to assign all the candidate nodes to a node for the establishment of an optimised routing table in terms of latency control. Each entry in a node's routing table has at least one prefix in common with the node's ID. Given the network size of 5000, maximum number of potentially qualified nodes for a node to build its routing table is approximately 5000/16 = 300 (parameter b = 4, hex number used for the nodeID). In other words, a central server needs to send at most 300 nodeID and IP pairs to each node for it to establish a routing table. Since each nodeID is a mere fixed-sized hex number, the cost for sending such information is negligible. When a node receives such information, it then selects from these candidates the ones that are closest to itself in terms of latency as entries of its routing table. In doing so, each entry in a node's routing table is also closest to the node among all the potentially qualified nodes in terms of latency. In the original Pastry design, a node builds its routing table through a 'probing approach' by contacting a nearby node and forwarding a joining

message. While this approach emphasizes a distributed approach, it does not guarantee an entry is the closest among all qualified nodes to the local node. It should also be noted that even in such a 'probing approach', a central service is assumed to be available to help a new node find a nearby node in the overlay.

Note that in the simulation setup, we have envisaged a hybrid approach by not eliminating a potential central service that is not expensive in network resources. In the context of P2P games, this is important from both the manageability and security perspectives. For details, please refer to discussion conducted in **Section 2.4.1**.

We then follow the design as discussed in Section 3.3. First, 127 peers are selected randomly as locale servers in the overlay. Then each node (player) will randomly pick a locale server to join. Again, a central service can provide a player with the IP and nodeID pair of such a locale server. When joining a locale server, a node routes a message with the key of the locale server's nodeID through the underlying Pastry distributed routing process. Through the joining process, Scribe multicast tree can be established on top of Pastry (see Section 3.2). We use the established multicast tree to disseminate state update information from locale servers (root of the tree) to the joined members. Our main objective is to investigate the performance of Scribe in distributing node stress in the overlay by considering the game traffic and practical peer's bandwidth capacity based on our game design.

Note that the choice of 5000 nodes for the simulation is to match the maximum capacity of a Lineage II central server to demonstrate P2P overlay scalability that is capable of supporting a large number of concurrent users without sacrificing major performance metric such as bandwidth and latency. If we aim to further increase the number of nodes in the overlay, e.g., to 10000 nodes instead of 5000, given our distributed game design in **Table 3.1,** 254 multicast trees will be established to simulate 254 locales, instead of 127 trees for 127 locales in an overlay of 5000 nodes. For each individual locale, given that the expected tree size remains the same regardless of the overlay size, we expect the bandwidth performance for the established tree will remain the same. However, if hop count is used as an indication of the potential latency performance, the expected hop count for an overlay of 10000 nodes is 3.32 ($\log_{16}1000$) , whereas in an overlay of 5000 nodes, the expected hop count is 3.07 ($\log_{16}5000$). The mere increase of 0.25 will not

have a significant impact on the results of our simulation with respect to latency study. Note that in Chapter 6, we use an overlay of 10,000 nodes to study the churn impact on P2P overlay in supporting our distributed game design, and further demonstrate the scalability of P2P overlay.

We ran the simulation 10 times and similar results were obtained in each instance. The results presented in the following sections are the sum of all the simulations.

3.4.2 Simulation Results

We present simulation results in the following sections. It should be noted that the simulation conducted in this chapter is static analysis of P2P system in supporting MMORPGs, in which we assume peers are playing in a particular part of the game world relatively statically. In Chapter 6, we conduct dynamic analsyis of P2P system in supporting MMORPGs, by studying churn behaviours of peers – the dynamic process of peer arrival and departure in the game world.

3.4.2.1 Fan-out

Our simulation establishes 127 multicast trees corresponding to 127 locales. We first measure the node stress in terms of fan-out, which is the number of children entries in each node. The results are presented in Figure 3.6.

It can be observed from **Figure 3.6** that, the common nodes that are *not* roots of any multicast trees, as shown on the left part of the figure, have less fan-out with nearly 90 percent of nodes having less than 5 children entries. The stress in terms of fan-out is mainly concentrated at the root nodes serving as the locale servers, for instance, as shown on the right part of the figure, more than 30 percent of root nodes have more than 20 children entries.

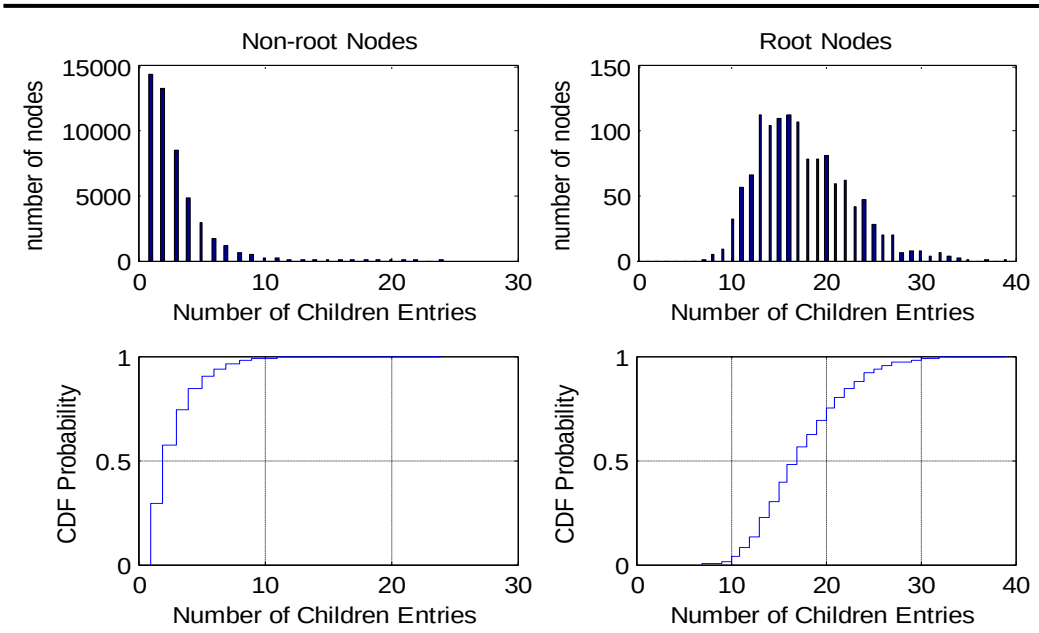
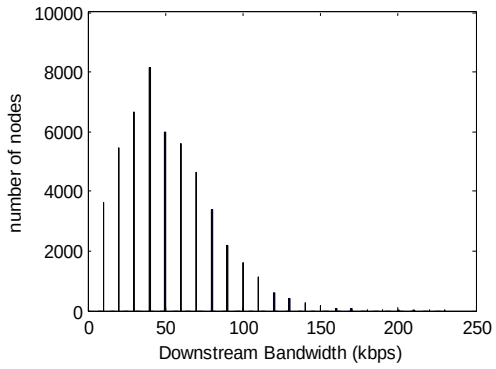


Figure 3.6 Fan-out for root and non-root nodes

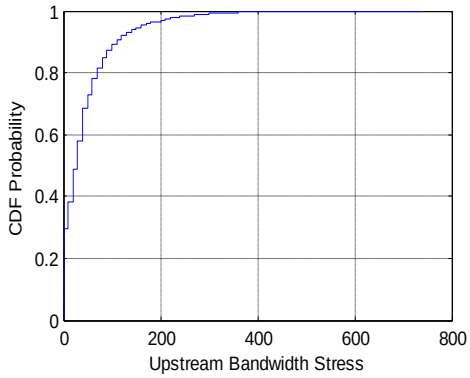
3.4.2.2 Bandwidth Usage

Given the peer-to-peer support for our game design, we further measure the bandwidth stress of nodes for disseminating state update information from the locale servers to the peers. A node incurs downstream bandwidth stress by taking traffic from its parents in the multicast trees it belongs to, and upstream bandwidth stress by disseminating the traffic to its children. The results are presented in **Figure 3.7**.

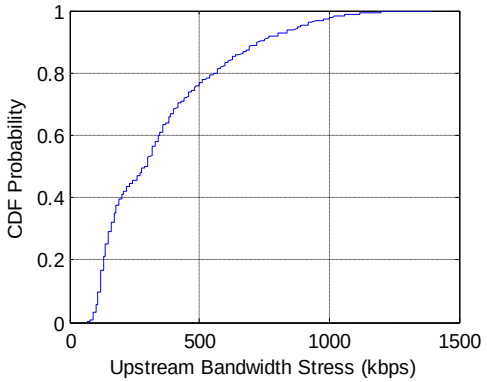
Histogram in **Figure 3.7** (a) shows that downstream bandwidth cost for peers is very low, with an average usage of 44kbps and maximum usage leas than 250kbps. Given a peer with a typical low-end DSL link of 1000/256kbps, the downstream bandwidth cost for supporting our game design is acceptable.



(a) Downstream Bandwidth Stress (kbps)



(b) CDF of Upstream Bandwidth Stress (kbps) – Non-Root Nodes



(c) CDF of Upstream Bandwidth Stress (kbps) – Root Nodes

Figure 3.7 Downstream and upstream bandwidth stress

However, **Figure 3.7** (b) and **Figure 3.7** (c) show that the upstream bandwidth cost is rather high. For non-root nodes, a long tail in the CDF **Figure 3.7** (b) shows that 5 percent of the peers are subject to an upstream bandwidth stress of more than 200kbps, with a peak close to 800 kbps. For root nodes, as shown in **Figure 3.7** (c), more than 50 percent of peers are subject to upstream bandwidth stress of more than 250kbps, with a peak close to 1500 kbps. This is obviously beyond a peer's upstream capacity since a common household end-host typically has less than 1000 kbp upstream bandwidth allocation, and hence is a bottleneck that has to be removed.

3.4.2.3 Removal of Bandwidth Bottleneck for the Non-root Nodes

We take two steps to remove the upstream bandwidth bottleneck revealed in the last section and constrain a peer's upstream bandwidth stress to below 256kbps. We first try to remove the bottleneck of non-root nodes by taking advantage of the flexible design of Pastry. In the path from a source node to its locale root node, if any nodes (aside from the root node) are overloaded in terms of upstream bandwidth stress, then the source node will try to take an alternative route towards the destination by picking a node in its neighbour set as the first hop. If an alternative path can be found to avoid the original congested path, then the cost would be an extra hop to a node in the neighbour set. An example is given in **Figure 3.8.** As demonstrated in Figure 3.8, if a relay node A in the multicast tree is overloaded in terms of the number of children entries, one of its children node B will choose to route to an node in its neighbour set first, in this example, node C. Then node C will route towards the root of the multicast tree to establish an alternative route path for node B, following Pastry prefix matching routing protocol.

Since the nodes in the neighbour set are close to the source node in the first place (see Section 3.2), the cost of an extra hop would be low in terms of latency. We present the distribution of upstream bandwidth stress for non-root nodes after applying this bottleneck removal algorithm in **Figure 3.9**. It is shown in **Figure 3.9** that the upstream bandwidth stress is effectively constrained to below 256kbps through this low-cost alternative path approach.

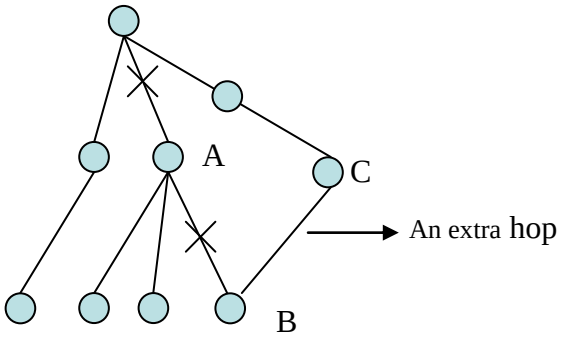


Figure 3.8 An alternative path example

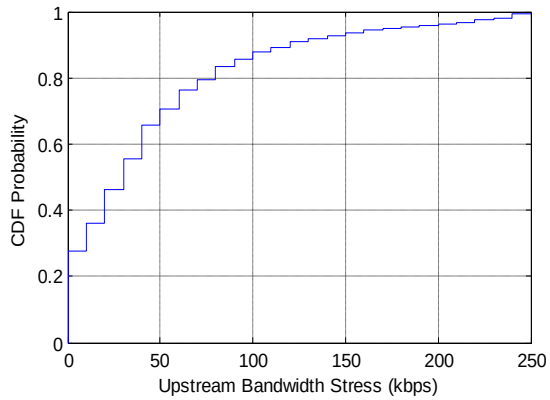


Figure 3.9 CDF of upstream bandwidth stress for non-root nodes after the alternative path approach

Note that this alternative path approach will not remove the bottleneck at the root nodes since the final hop of any paths will always be toward the destination root node.

3.4.2.4 Removal of bandwidth Bottleneck for the Root Nodes

We then take the second step to remove the upstream bandwidth stress bottleneck at the root nodes of the multicast trees that serve as locale servers. We utilized an algorithm similar to the one proposed in the original Scribe paper that works as follows. When a root node detects that it is overloaded, it then works out how many children need to be dropped given its upstream bandwidth capacity and state update bandwidth requirement. Those dropped nodes will then try to connect to the children of the root node, which are less likely to be overloaded. For instance, a Map Type 4 (**Table 3.1**) needs a bandwidth requirement of 40kbps for disseminating state update information for each connection.

Hence, a root node for a Map Type 4 multicast tree can have at most six children entries if we assume conservatively a peer with low-end upstream bandwidth capacity of 256 kbps (6 x 40 = 240kbps < 256kbps). If more than six children are detected by such a root node in its children table, it performs the following operations: 1) it selects the most stressed nodes in terms of upstream bandwidth usage and then drops them from its children table; 2) the dropped node scans through the retained children entries of the root node and selects one that via which it can reach the root with minimum distance in terms of latency; 3) the dropped node joins the selected node and becomes a child of the selected node in the corresponding multicast tree. This algorithm can be executed recursively until node stress constraints are met. For instance, if a root node's retained children all have met their upstream capacity and can no longer take any child, a dropped node from the root can search through the children of a root node's retained children entries, following the same algorithm described above, until an appropriate parent is found. In all the ten simulation instances conducted, it only took a single recursion for a dropped child to find a new parent to connect to. **Figure 3.10** shows that the upstream bandwidth stress of root nodes is effectively constrained to below 260 kbps after applying the algorithm discussed above.

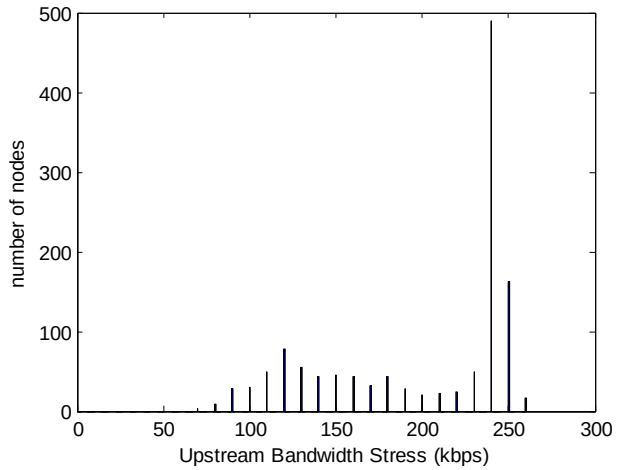
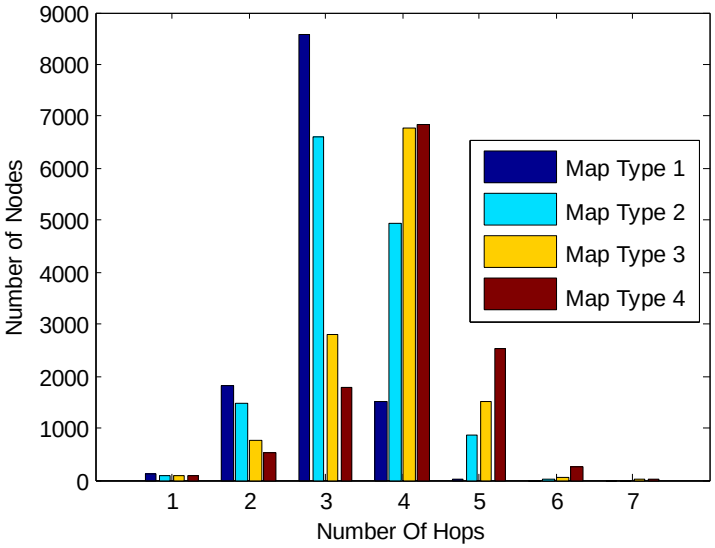


Figure 3.10 Histogram of upstream bandwidth stress of root nodes - after the bottleneck removal

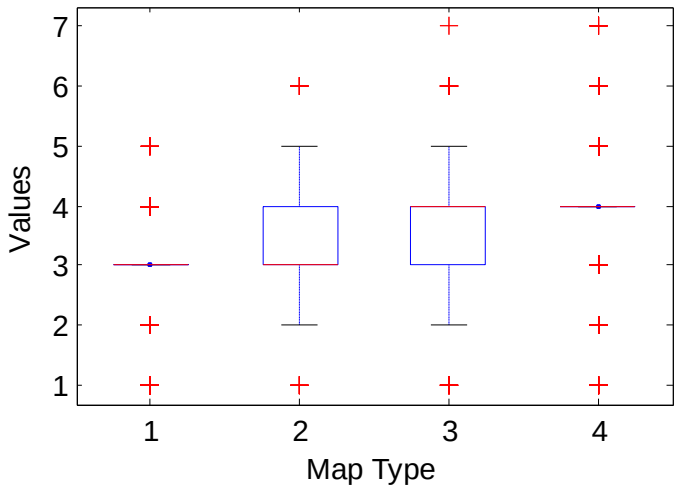
3.4.2.5 Cost of Bottleneck Removal

We finally investigate the cost of bottleneck removal algorithm discussed above. In the case of removing bottleneck for non-root nodes, the cost is an extra hop for a node to reach the root of the multicast tree through the alternative path approach (see Figure 3.8). In the case of removing bottleneck for root nodes, the cost is an extra hop or hops as a result of the recursive algorithm**. Figure 3.11** (a) shows the number of hops vs. the number of nodes for each type of the maps in our game simulation. **Figure 3.11** (b) is the box plot of the number of hops for each Map Type.

Figure 3.11 (a) and (b) show that, while the state update bandwidth requirement increases from 10kbps (Map Type 1) to 40kbps (Map Type 4), i.e., four times, the hop number shifts slowly to the right from Map Type 1 to Map Type 4, and the average number of hops only increases from 3 to 4. While there are a small number of maximum hops of 5 for Map Type 1, nearly ninety percent of the hops are less than 3, which justify our map design in terms of latency control for supporting real-time sensitive interactivities. There are a small number of nodes that are subject to a maximum of 6 to 7 hops that is the result of the bottleneck removal algorithm. This might require a map design that supports less real-time sensitive interactivities, which is, in turn, an unavoidable cost for being able to support more players, as in the case of our proposed interactivities and capacity for Map Type 2 to 4.



(a) Histogram of Number of Hops



(b) Box plot of Number of Hops

Figure 3.11 Hop counts after bottleneck removal

3.5 Conclusions

In this chapter, we have presented a distributed game design that supports Massively Multiplayer Online Role Playing Games (MMORPG) through a Pastry overlay to address the scalability issues found in current pre-dominant Central Server Model. The design considers the following aspects. First, a typical MMORPG traffic pattern has shown that there is a substantial asymmetry between the downstream and upstream bandwidth stress. Hence, it is necessary to use a multicast tree to disseminate game state update traffic in order to save bandwidth at the server side. We have used Scribe to establish multicast trees for this purpose. We then take advantage of the interest management technique used in MMORPG by organizing peers with a common *area of interest* into a common multicast tree. This is to take advantage of a natural match between the self-organizing property of MMORPG and the self-organizing characteristics of a peer-to-peer overlay. We further conducted a traffic trace to establish actual state update bandwidth usage in relation to the number of players that can be supported in a locale. Together with latency consideration, we envisaged the size of the multicast trees for the distributed game design to closely reflect an advanced MMORPG configuration.

Results have shown that Scribe trees perform well in distributing bandwidth stress among peers. However, there exists a bottleneck in upstream bandwidth stress, which can be removed by low-cost algorithms. Furthermore, the results have shown that Scribe tree scales well in terms of tree length. When a tree size increases five times, the length of the tree only increases less than two levels. This property contributes to latency control, which is critical in determining the nature of interactivities that can be supported in MMORPGs.

Chapter 4. Enhancing the Multicast Performance of Structured P2P Overlay in Supporting MMORPGs

Abstract

Scribe is a scalable application level multicast infrastructure. We have developed two techniques in this Chapter to improve the performance of Scribe in terms of latency and bandwidth distribution. The first technique identifies that the final hop of Scribe traffic path is largely selected without any proximity consideration and incurs the longest distance travelled. To overcome this, we introduce Proximity Neighbour Selection (PNS) into the final hop for latency improvement. The second technique builds a hierarchical two-level overlay. While PNS can be applied at both levels for latency performance, the bandwidth stress required by applications can now be distributed among the nodes in the higher level overlay. Our simulation using GT-ITM topology has shown that both techniques have improved the latency performance by more than 30 percent, and the two-level overlay has improved the bandwidth distribution by up to 2.7 times, comparing with what can be achieved by a standard Scribe overlay.

We have developed the techniques in the context of Massively Multiplayer Online Role Playing Games (MMORPGs). While Scribe provides a possible platform for the scalable deployment of MMORPGs, game developers may leverage the proposed techniques to enhance the design of real-time interactions between players in the game world.

4.1 Introduction

In the preceding Chapter, we have presented a distributed game design based on Scribe to multicast game traffic. When studying the performance of Scribe, we have focused on bandwidth distribution and latency, the two metrics that have significant impacts on game features and quality from network perspective. In this Chapter, we take a step further by plugging in an internet topology model into our simulation package. Through quantitative analysis, we have identified a couple of potential bottlenecks of Scribe in supporting our game design. Those bottlenecks, in turn, have led to the development of two important techniques in this Chapter to enhance the performance of Scribe in supporting our game design.

In recent years, there has been much interest in peer-to-peer (P2P) multicast in response to the impracticality of the deployment of network-layer IP multicast. P2P multicast organizes end hosts (peers) into networks overlaid on the Internet that functions as a communication platform. Multicast related features such as member management, and multicast tree building are implemented at peers.

Scalability and performance are two critical concerns for P2P multicast design. Scalability is closely related to the control and maintenance overhead required by the P2P overlay management protocol. Performance refers to the metrics such as bandwidth distribution and routing latency of the P2P multicast tree established.

Scribe is a P2P multicast design that is developed on top of Pastry, a structured P2P overlay. Scribe achieves scalability by leveraging the underlying Pastry substrate that adopts a decentralized P2P model through the use of Distributed Hash Table (DHT). As a result, Scribe is scalable with respect to supporting large numbers of multicast groups, with potentially large number of members in each group.

With respect to latency performance, Scribe adopts a light-weight technique called Proximity Neighbour Selection (PNS) to improve the latency associated with multicast communication. The traffic path in a P2P overlay is through overlay hops from one peer to another. PNS, in simple terms, selects peers that are topologically close to each other

to form the traffic path. The requirement for bandwidth distribution in P2P multicast is an application specific issue. Due to the random nature of DHT, some peers may be subject to heavy stress in a large-sized Scribe tree. We have demonstrated in the previous chapter (Section 3.4.2.3 and 3.4.2.4) that a light weight bottleneck removal algorithm can be used to shed the potentially heavy stress.

In this chapter, we continue the study of Scribe in the context of Massively Multiplayer Online Role Playing Games (MMORPGs). MMORPGs are traditionally deployed in a Central Server model. Central Server model achieves scalability through partition techniques. A game world is partitioned into multiple locales, and each locale is processed by a locale server, typically in a server cluster. A locale may be further partitioned into multiple *areas of interest* (AOI), and a player only receives from the locale server a subset of the *state update flow* that represents the latest states of the objects in his AOI. It should be noted that when a large number of players connected to a central server cluster, while the computation can be distributed, the possible congestion of bandwidth stress at the server can not be avoided. Furthermore, the resources have to be over provisioned to handle peak loads.

On the other hand, the fundamental attractiveness of MMORPGs lies in the design of real-time interactions between players. In the context of MMORPG, the quality of interaction can be generally measured by two metrics. The first is the population of avatars within an AOI that is defined as the number of players who can interact with each other simultaneously. The second is the responsiveness of the game to player actions, which is defined as the time difference between when an action is issued by a player and when the action is executed and recognized by other players in the game world. As revealed in Section 4.2 in this chapter, the two interaction metrics are directly influenced by the bandwidth usage for state update flow and network latency respectively. As a result, the current game design of interactions is largely constrained by the bandwidth shortage and high network latency.

However, given that peers are usually located at the edges of the networks and have limited bandwidth capacity, the performance study is of paramount importance to justify the use of Scribe in supporting applications that are both bandwidth and latency intensive. In this chapter, we argue that a standard Scribe setting is not optimal with

respects to latency performance and bandwidth distribution. This is particularly true in the context of MMORPGs, in which the fundamental design concern is to improve the interaction metrics for enhanced player experiences, and hence requires significant resources such as bandwidth and low latency level. As a result, while Scribe presents the possibility of providing a platform for the scalable deployment of MMORPGs, the challenge is to further improve the performance of Scribe and, at the same time, maintain the integrity of Scribe as a structured P2P overlay.

To that end, we have developed a couple of techniques in this chapter to enhance the multicast performance of Scribe in supporting MMORPGs. The key to our techniques is the recognition that while the build-in Proximity Neighbour Selection of Scribe can effectively reduce the latency associated with the initial overlay hops of a traffic path, the final hop of the traffic path is largely selected without any proximity consideration, and usually is a major contributor to the overall delay. As a result, demonstrated by the simulation results, both of our techniques can improve the latency performance by more than 30 percent, and the second technique further improves the bandwidth distribution by a factor of 2.7, comparing with what can be achieved in a standard Scribe setting.

Moreover, in considering the actual traffic pattern of advanced MMORPGs, we have demonstrated that, at the cost of practical peers' capacity, the techniques can promote the interaction metrics to a high level, which would otherwise incur a higher cost to a standard Scribe setting with respect to latency, or an even worse prohibitive cost to a central server with respect to bandwidth usage.

The rest of the chapter is organized as follows. In section 4.2, we reveal how bandwidth and latency impose constraints on the interaction metrics in MMORPGs. In section 4.3, we briefly review the fundamental ideas of game designs using Scribe to achieve deployment scalability, which is an inherent issue faced by current Central Server model. In section 4.4, we present the techniques to improve the multicast performance of Scribe in supporting the distributed game design. We discuss related work in section 4.5 and conclude in section 4.6.

4.2 Current Interaction Features in MMORPGs

MMORPGs distinguish themselves from other genres of online games by allowing a large number of players (usually in the range of thousands) to share a common game virtual world constituted by a huge map. Popular MMORPG titles include EverQuest, Lineage and World of WarCraft. Those games boast millions of registered users and thousands of concurrent players in a common game virtual world.

The traffic trace in Lineage II has shown that, although the bandwidth usage for a single connection in Lineage II is minimal because of using the technique of AOI, when a large number of players are connected simultaneously to a central server, the bandwidth required by the state update flow has become a major cost concern for the game providers. For detailed analysis of the game traffic trace, please refer to Section 3.3.1 in Chapter 3.

The state update flow is closely associated with the number of objects in an AOI and the state update rates for those objects. In Section 3.3.1 in Chapter 3, we have further conducted a traffic trace at the client side for World of WarCraft (WOW), which is arguably the most popular MMORPG on the market at present. The traffic trace has shown that the bandwidth usage for the state update flow is approximately linearly correlated with the number of players in a typical AOI.

Based on these traffic traces of MMORPGs, it can be inferred that bandwidth is the key component that restricts the interaction metric with respect to the population level of an AOI. For instance, it was observed in the traffic trace that the maximum population in a typical AOI in the game world is approximately 30 players when the bandwidth usage for the state update flow reaches approximately 15kbps per client. Hence, if the population level of an AOI is increased to 200 players, the projected bandwidth usage for the state update flow would reach 100kbps for each player in the corresponding AOI. Subsequently, this would result in an upstream bandwidth stress of 20Mbps (100kbps x 200) at the server side to be able to support this single AOI, assuming a Central Server model. Considering a game universe consisting of many (e.g., in the range of hundreds)

AOIs, the cost of such a communication load would be prohibitively high for the game provider.

We envisage that the increase of the population level of an AOI could bring many novel features into MMORPG design to improve the game playability. For instance, a large scale battle field involving an army against another army, a social congregation of hundreds of participants, all require a large number of players who can interact with each other concurrently, and the challenge is to overcome the potential bandwidth bottleneck.

The other interaction metric is the responsiveness to player actions. From the player's perspective, this metric is generally perceived as the response time of the game to actions. In current Central Server model, the actions issued by players are mainly executed by the central server. After the execution of the players' actions, the server then sends a list of objects (state update flow) to the players for rendering a new state of the world as the results of the actions. To tolerate the latency heterogeneity of the Internet, current MMORPGs usually only support slow-paced actions that do not require quick responsiveness. For instance, fighting in MMORPGs is usually slow-paced and involves medieval weapons that do not require quick physical responses from the players. Moreover, the results of the fighting are largely predetermined by the game logic according to the level of strength of the avatar models or through somewhat random operations or factors. Consequently, MMORPGs can usually tolerate up to several hundreds of milliseconds or even seconds of latency without severe impact on player performance (whether winning or losing). Nevertheless, the quality of game playability with respect to action responsiveness could be significantly influenced by a high latency[56]. On the other hand, the latest development trend is to have fast-paced (quick responsiveness) actions integrated into MMORPGs that traditionally only support slow-paced actions. For instance, BigWorld [57] is a server technology that has this design potential as one of its principal aims.

To conclude, with respect to the interaction metrics, current MMORPGs have a small population of players in an AOI (e.g., tens of players) who can interact with each other concurrently, and slow action responsiveness. The constraints of the interaction metrics are directly imposed by bandwidth shortage and high network latency.

4.3 Distributed Game Design Using Scribe

Distributed game design may leverage the partition techniques employed in current MMORPGs. While being huge in size, a game virtual world is constituted of many loosely linked locales. Hence, P2P system provides a possibility to have resources needed for supporting those multiple locales distributed among peers.

To the best of our knowledge, Kuntsson B. et al. [48] first proposed to marry MMORPG with structured P2P system for both having the property of self-organizing. It should be noted that in [48], the population level of an AOI is relatively small with 10 players in an AOI.

In Chapter 3, we extended the design idea by considering the advanced MMORPG settings. The use of Scribe is justified by considering the significant asymmetry between the client generated traffic (action commands) and the server generated traffic (state update flow). It is proposed to use Scribe to multicast game state update flow in order to save bandwidth usage and promote the population level of an AOI. However, action commands can be unicast to the root of the multicast tree due to the low bandwidth usage incurred. Our proposed traffic model is demonstrated in **Figure 3.3** in Chapter 3. As demonstrated in the traffic model, the root of a multicast group serves as a server for an AOI. Multiple AOIs can form a huge virtual world taking advantage of the scalability nature of the structured P2P overlay.

Furthermore, given the limited capacity of peers, the design considered a trade-off between the two interaction metrics. This is because a large-sized multicast group corresponding to a high population level of an AOI would usually incur a heavy stress at the root of the multicast tree, due to the random nature of Scribe. As a result, a "bottleneck removal algorithm" is usually required to remove the stress at the cost of latency performance.

The simulation conducted in Chapter 3 based on a Pastry overlay of 5000 nodes has shown that, while the size of a multicast group increases from 20 to 100 members (the population level of an AOI = 20 and 100 respectively) that is a five fold increase, the

average overlay hops only increases from 3 to 4. Note that the traffic required by the state update flow for 20 and 100 member multicast group are 10kbps and 50kbps per-connection respectively, projected from World of Warcraft traffic trace. P2P multicast using Scribe effectively constrained the upstream bandwidth usage of peers in the overlay to below 260kbps (a low-end ADSL configuration) in supporting such a traffic required by the projected state update flow.

To make the obvious point, to support a population level of 100 players in an AOI, a Central Server model would incur an upstream bandwidth usage of 5 Mbps (50kbps x 100 = 5 Mbps) at the server for the state update flow, whereas Scribe can build a tree with *the length less than four* to support the same level of population and only requires low end ADSL bandwidth capacity of peers.

It should be noted that while the effectiveness of Scribe in saving bandwidth is demonstrated in Chapter 3, the latency performance relies on the small number of overlay hops achieved by the Scribe routing protocol.

4.4 Enhancing the Multicast Performance of Scribe

For the distributed game design presented in Chapter 3 and summarized in the previous section, we can assert that Scribe scales well in terms of bandwidth distribution and latency performance *provided we are assured that a small number of hops would be an indication of low latency*.

However, it is a general concern in the research field [55, 58, 59] that the uniform distribution of NodeID in Pastry could lead to long hop distances that negates latency performance. In addition, large-sized Scribe multicast tree may have nodes subject to excessive stress. In this section, we first discuss the multicast performance of a global Pastry overlay. We then present techniques to enhance the multicast performance of Scribe. We present simulation procedures and results along the way in assisting the discussion.

4.4.1 Experimental Setup

We used Georgia Tech. GT-ITM tool to generate a transit-stub (TS) topology for experimental purposes. In this topology, the nodes are grouped into either transit or stub domains. In our experiment, there are three transit domains with an average of eight nodes (transit nodes) in each domain. Each transit node is connected to three lower level stub domains on average. There are on average eight nodes (stub nodes) in each stub domain. TS topology emphasizes the locality feature of the Internet. To this end, link weight is assigned such that traffic path between two nodes in the same domain will remain within that domain [60]. In [61], a similar topology is used to model the Internet. We then have 15 end hosts on average connected to each stub node in the topology to model an overlay network of 10000 end hosts. We use the topology assigned link weight as RTT for the links. The link between an end-host and a topology node is assigned a random RTT value between 3 – 10 ms. For detailed configuration of GT-ITM, please refer to Appendix B.

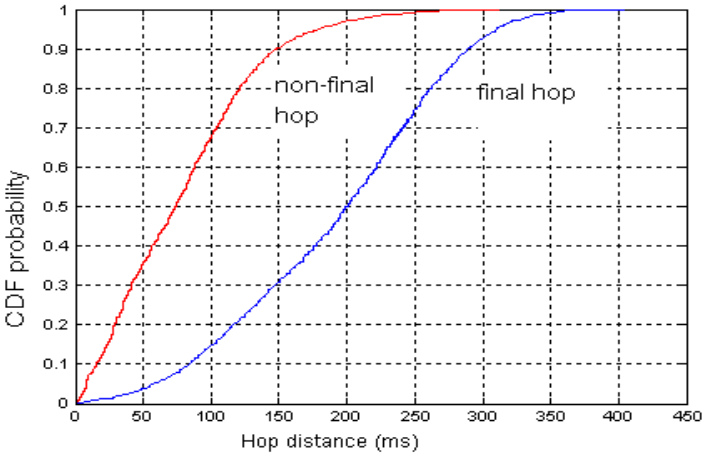
4.4.2 Multicast Performance of a Global Overlay

With respect to latency, while Pastry achieves good performance in terms of controlling the number of routing hops, Proximity Neighbour Selection (PNS) is a light-weight technique used by Pasty to further control the latency of each hop. When a Pastry node has multipe candidate nodes as entries for its routing table (see **Figure 2.4**), the node will ping each of the candidate nodes in the overlay, and selects one that is closest to itself in terms of proximity metric, such as latency, as the entry for its routing table. In receiving a message by a node, if the next hop for the message is an entry in that node's routing table, the hop distance can obviously be controlled depending on the number of available candidate nodes in constructing the routing table entry. In other words, the more candidate nodes exist, the more chance a node can find a closer entry for its routing table. However, the number of candidate nodes for a routing table decreases exponentially as the level of the routing table increases.

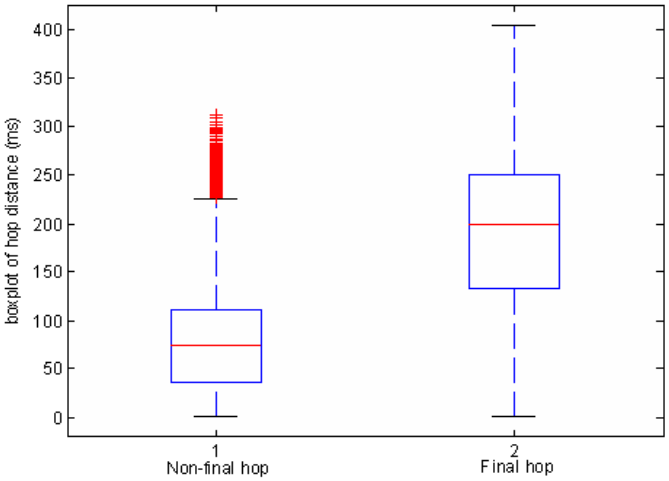
The first level of a node's routing table contains nodes whose IDs share one prefix with that node. For an overlay of size 10000, if hexadecimal IDs are used, a node would expect to have 10000/16 = 666 available nodes to choose from as entries for its first

level routing table. The second level of a node's routing table has nodes whose IDs share 2 prefix digits with that node. As a result, the node would expect to have $1000/16^2 = 42$ nodes available to choose from as its second level routing table entries. This exponential reduction of candidate node size would result in gradual deterioration of the PNS effectiveness and increased hop distance along a traffic path.

To investigate the effectiveness of PNS, we distinguish a final hop from a non-final hop for a traffic path form source node to the destination node. While PNS may effectively reduce the distance of the initial hops of a traffic path, the final hop towards the destination node is usually routed towards an entry in a node's *leaf set* in a topologically randomized manner without the application of PNS. To demonstrate this, we established a global Pastry overlay of 10000 nodes, and **Figure 4.1** shows the Non-final hop distance vs. the final hop distance.



(a) CDF of Final Hop Distance



(b) Box plot of Hop Distance

Figure 4.1 Non-final hop distance vs. final hop distance

It is shown in **Figure 4.1** (a) and (b) that the distance of the final hop dominates the total distance of a traffic path. For instance, as shown in **Figure 4.1** (a), given our topology, there are 10 percent of non-final hops compared with 70 percent of final hops that are experiencing a latency level of more than 150 ms. **Figure 4.1** (b) further reveals that general statistical metrics such as median value, first quartile and third quartile of the hop distance all increase more than twice from non-final hops to final hops.

We hence envisage that to improve the latency performance of Scribe, the key may rely on further improving the final hop latency of the traffic path in Scribe.

4.4.3 Apply PNS to the Final Hop

4.4.3.1 Motivation and Approach

Clustering technique is a widely discussed topic in the research field [55, 58, 59]. The fundamental idea is to find nodes that are close to each other, and provide this locality information to applications to achieve better performance. For instance, Ratnasamy et al. [55] proposed to use the landmark scheme to have nodes clustered into bins. In this scheme, the landmarks are a number of well known nodes in the Internet. A peer in the overlay can ping those landmarks to obtain its topological coordinate. Peers with the same coordinates are then clustered into the same bin. Simulation through a Transit-Stub topology, a topology similar to GT-ITM that is used by us for simulation, showed that the RTT between nodes within the same bin is on average 4.06 times shorter than the RTT between nodes across bins.

The use of such a binning technique is apparent for applications involving content distribution. Popular contents can be cached into a local bin so that local peers can have access to that content with better use of resources (e.g., less access time, no more traffic required across the bins).

However, for distributed game design, the simple caching scheme will lead to multiple decision points that involve extra complexity. To illustrate, let us assume nodes are effectively clustered into bins and the state of the game world managed by a single AOI server is distributed among multiple peers in the local bins as shown in **Figure 4.2**.

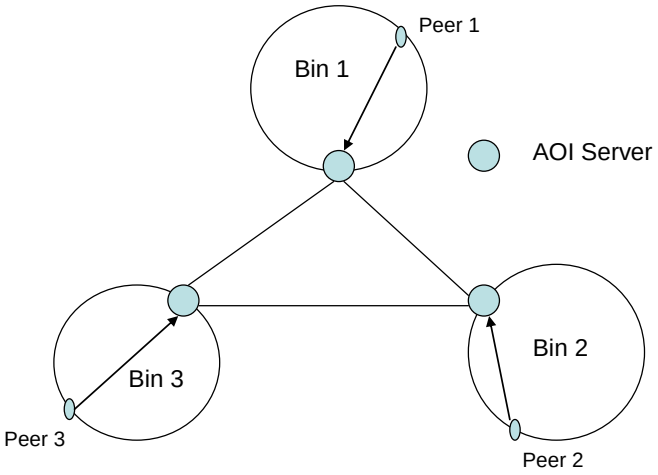


Figure 4. 2 Distributed AOI Servers

In **Figure 4.2**, a single AOI is managed by three distributed servers for that AOI to improve local peers' access time to the copy of the game state. To maintain the consistency of the game state, synchronization is now needed between the three AOI servers so that the three players can have a relatively common view of the state of the game world. For instance, if peer 1 takes an action, the result of the action can not be decided by the AOI server in bin 1 alone, but by all the AOI servers for the computation of an update of the game state. Hence, the multiple decision points will possibly lead to extra computation and potential complicated synchronization techniques required. For a detailed investigation into those issues, please refer to [56].

We envisage that maintaining a single decision point for an AOI can avoid the complicated issues involved in distributed server design. Moreover, since AOI level partition is used by most game engines, the porting of games from server clusters to a P2P overlay will be more practical from the software engineering perspective.

To maintain a single decision point (A single AOI server), if the AOI server is located in a bin, the challenge is to find a short path between a peer in a different bin and the AOI server, while maintaining the integrity of Pastry overlay. We have discussed in the previous section that PNS is a light-weight technique in controlling distance of initial hops of Pastry traffic path, and we have identified that the final hop dominates the distance of the total traffic path because PNS does not apply to the final hop of Pasty

traffic path. With those in mind, we here propose an approach to apply PNS to the final hop as follows.

First, instead of establishing a global P2P overlay, through clustering techniques, multiple local overlays can be established for better use of resources. Moreover, a major concern over Pastry design is the possibility of undesired hops due to the random properties of the design. For instance, a path might traverse across bins before coming back to a nearby node in the local bin. By clustering nodes, this possibility is eliminated because the traffic path is constrained to local links. In this thesis, we rely on available techniques for node clustering. However, we envisage that game distributed design might have specific needs for clustering techniques. For instance, while the landmark binning scheme can effectively group nodes that are close to each other, the number of bins can not be easily controlled. As a result, when the number of landmarks increases, clustering heuristic may generate too many bins with each bin having only a few nodes in it. We will leave it to future work to investigate game specific needs for the design of clustering heuristics.

Second, when peers in a local bin connect to an AOI server located in a different bin, instead of connecting to it through the global overlay routing, peers are routed to a local relay node that is close to the AOI server node as shown in **Figure 4.3**. Note that in doing that, the hop from the relay node to the AOI server node becomes the final hop of the traffic paths for peers.

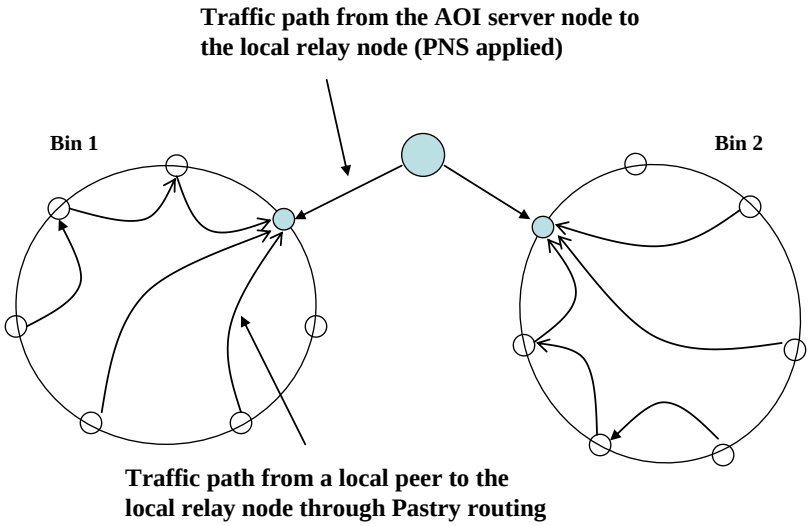


Figure 4. 3 PNS Applied to the Final Hop

We apply PNS to the choice of the relay nodes as a light-weight technique to improve the final hop latency performance. When selecting a relay node in a local overlay, an AOI server node can ping a number of nodes in that local overlay and pick the closest as the relay node. In reality, a combination of different metrics may be used for the choice of a relay node, such as bandwidth, latency and computation power. We choose latency as the metric because our intent here is to promote the responsiveness to actions for MMORPGs.

Recall that the final hop of a traffic path in a global Scribe is largely selected at random. When two nodes are located in different domains, the final hop usually traverses a link across domains. By applying PNS to the final hop, we are actually exploiting network proximity at the cross domain link. For instance, among potential nodes that could become the relay nodes for the final hop, some may be situated near network access point, such as getaways to administrative domains. The choice of such nodes as final hop relays would exploit with best efforts a shortcut that travels through a link across domains. In a global Scribe overlay, PNS only applies to the initial hops of a traffic path, which more likely travel across links within a domain. Considering the final hop across domains usually incurs the longest distance travelled, the application of PNS to the final hop in the overlay routing path should be more effective in reducing the total distance travelled.

4.4.3.2 Simulation Procedure and Results

To test the effectiveness of the final hop PNS, we developed a simulation procedure as follows. First, we take advantage of the locality properties of the TS topology by clustering nodes based on which domain they belong to. Based on transit domains, we have three clusters of nodes (recall that we have three transit domains in our topology model). To achieve finer granularity, we further group nodes based on which stub domain they belong to. We are able to have 24 clusters of nodes because there are 24 stub domains in our topology model. For transit domain level clustering, we have 3334 end-hosts on average in each domain. For stub domain level clustering, we have 420 end-hosts on average in each domain. Our intention is to compare the performance of simulation between different levels of granularity in terms of topological awareness achieved. For instance, transit domain level clustering achieves less topological awareness than stub level clustering does. In other words, the average distance between a pair of nodes in a transit domain is higher than that in a stub domain.

Second, a local structured P2P overlay is established following the Pastry design. It should be noted that for a node to join a local overlay, the node only needs to contact a node that is already in the local overlay to be bootstrapped into the overlay network. Aside from this, the procedure of establishing a local overlay makes no difference from that of establishing a global overlay.

Finally, we pick randomly a pair of end-host nodes in the topology, one as the source node and the other as the destination nodes (a potential AOI server). We compare the distance travelled from the source to the destination between three settings: 1. the global overlay; 2. three clustered overlays with final hop PNS applied; 3. twenty-four clustered overlays with final hop PNS applied. The results are presented in the following figures.

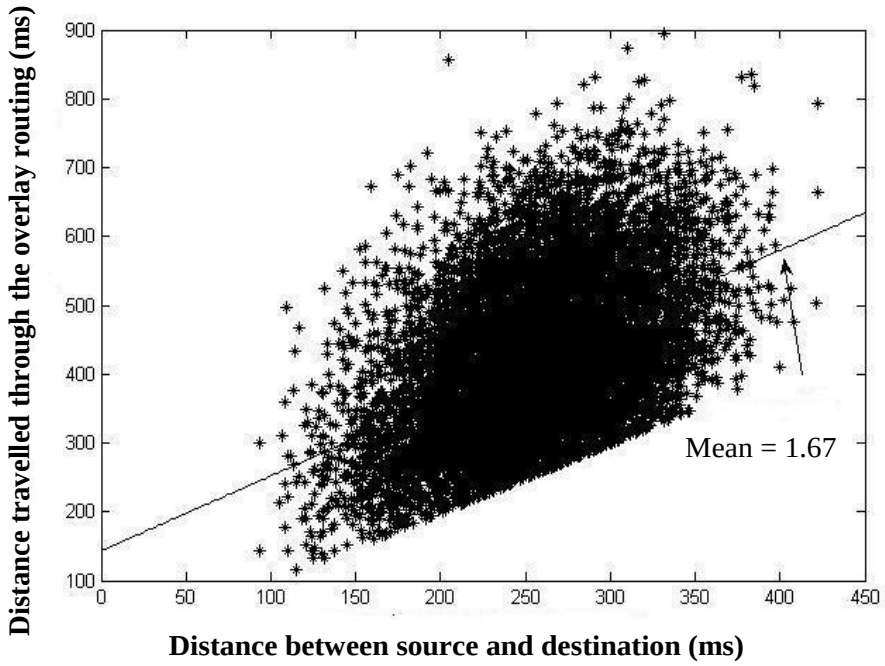


Fig. 4.4(a) Global Overlay Setting

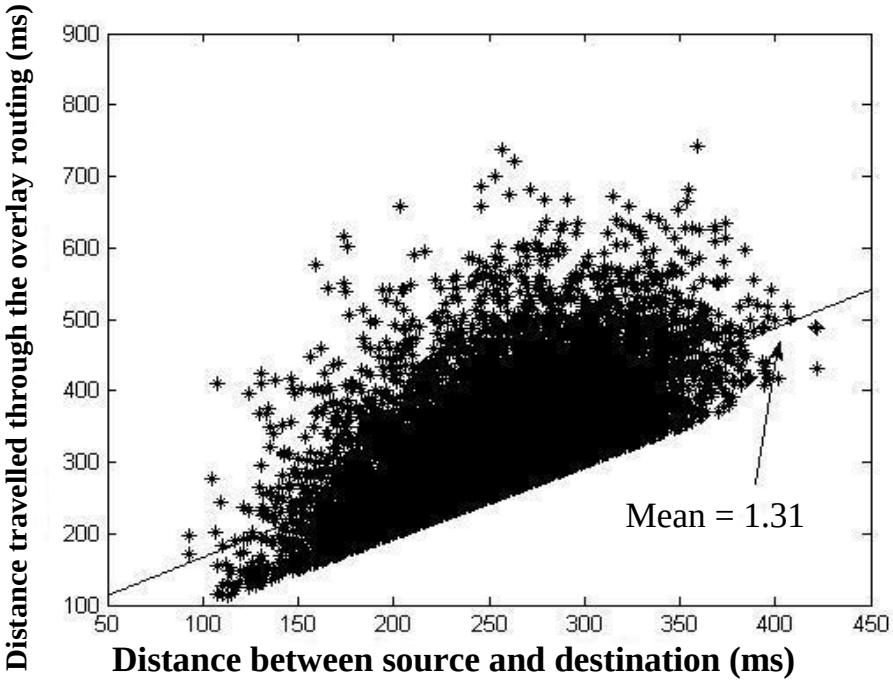


Fig. 4.4(b) 3 Clustered Overlay Setting

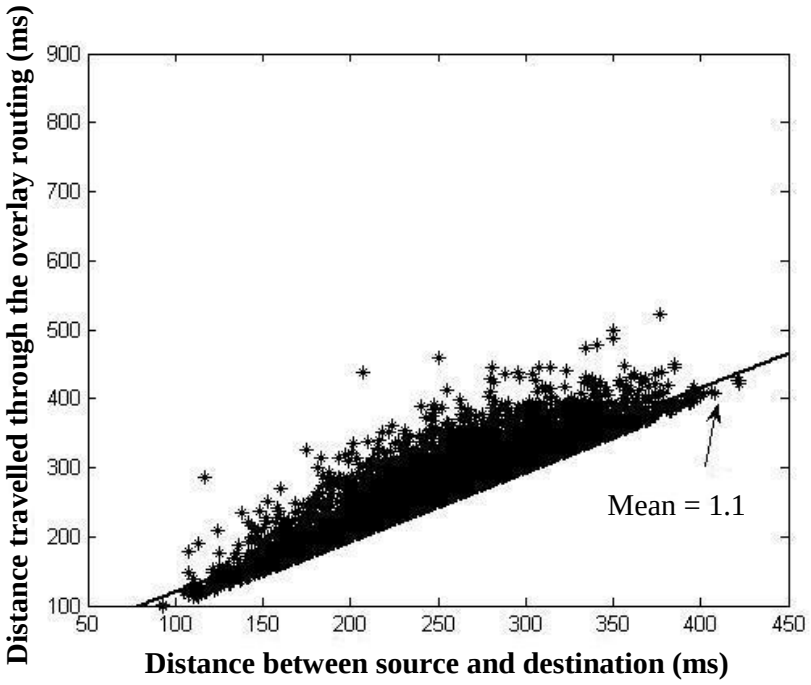
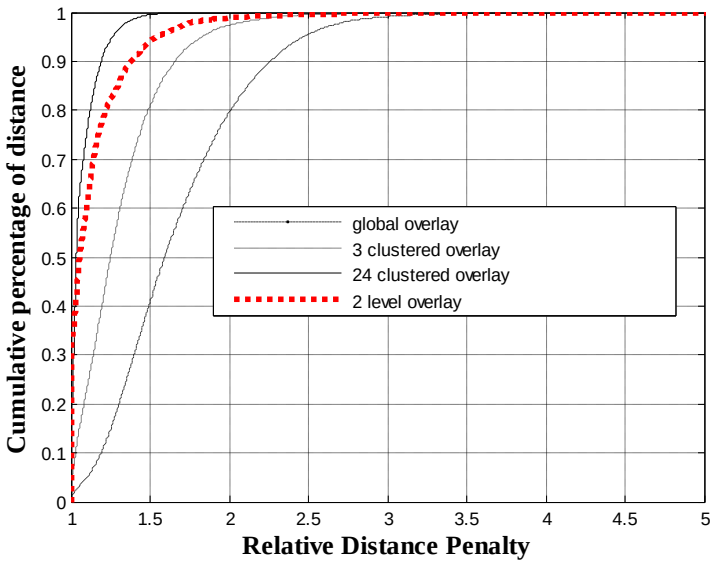
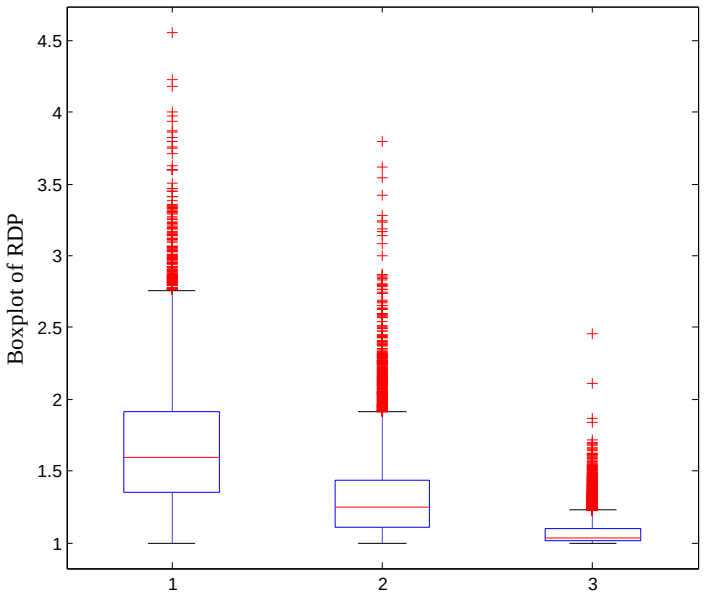


Fig. 4.4(c) 24 Clustered Overlay Setting

Figure 4. 4 Topology distance vs. overlay routing distance



(a) CDF of RDP



1. global overlay 2. Three clustered overlay 3. Twenty four clustered overlay

(b) Box plot of RDP

Figure 4. 5 Comparison of Relative Distance Penalty (RDP)

The scatter plots in **Figure 4.4** (a) (b) (c) show the overlay routing latency vs. the shortest path latency for the three settings respectively. The shortest path latency on the horizontal axis is obtained via Dijkstra algorithm. From **Figure 4.4** (a) to (c), it can be observed that the best fit line representing the mean value of the sample points has decreased from 1.67 in the global overlay, to 1.31 in 3-clustered overlay, and further down to 1.1 in 24-clustered overlay, suggesting the overlay routing performance improves significantly as the level of granularity for clustering increases. It should also be noted that from **Figure 4.4** (a) - (c), the sample points have become less scattered and more concentrated around the best fit line, suggesting the performance improvements are consistent.

Figure 4.5 (a) and (b) compare Relative Distance Penalty [47] of the three overlay settings. RDP is defined as the ratio between the overlay routing distance and the shortest path distance using the underlying topology, and serves as an indication of the overlay routing performance. **Figure 4.5** (a) shows that, while 10 percent of the routing distance has achieved RDP of 1.2 in the global overlay, approximately 30 percent of the routing distance in the 3-clustered overlay setting have achieved the same RDP level. The percentage of this RDP level has further increased to 90 percent in the 24 clustered overlay setting. The average RDP of the 24-clustered overlay setting is 1.1. In other words, it is nearly as good as the shortest distance achievable. The curve labelled as "2 level overlay" will be discussed in the next section. The box plot in **Figure 4.5** (b) further reveals the general statistical metrics for the three settings. In addition to the general improvement of those metrics up to 30 percent as the level of topology awareness increases, a consistent drop of the tail is evident.

To conclude the simulation results, the techniques of clustering and the application of PNS to the final hop can significantly improve the latency performance of P2P multicast using Scribe. Game developers could take advantage of these techniques to improve the responsiveness of MMORPGs.

4.4.4 Two-Level Overlay

4.4.4.1 Motivation and Approach

We further consider the population level of an AOI that may cause bandwidth stress on AOI server nodes. Recall that an AOI server node connects to peers in a local bin through a relay node according to the technique discussed in the previous section. Hence, an AOI server node would have a number of direct connections (number of children entries or fan-out that determines bandwidth stress) to relay nodes that is equal to the number of bins clustered in the worst case. Note that the fan-out of a relay node can be controlled by the Scribe tree established by peers within the local bin, but the fan-out of an AOI server node is relatively fixed (**Figure 4.3**).

The inflexibility of the bandwidth stress at the AOI server node could create a potential bottleneck depending on the desired population level of an AOI. For instance, the 24 clustered overlay setting has achieved the best latency performance in our simulation but would result in a fan-out of 24 at the AOI server node, which is the root of the multicast tree for the state update flow. For a quick back-of-the-envelope calculation, if an AOI server node is to support a population level of an AOI of 200 players, each direct connection of that server node would be subject to an upstream bandwidth stress of 100 kbps (Section 3.3.1), estimated from the World of WarCraft traffic traces, and result in a total bandwidth stress of 2400 kbps at the server node. Such a stress could be expensive and discourages a client computer (a peer) to take the role of an AOI server. Moreover, it will be sometime before the main stream ADSL setup can achieve that upstream capacity.

We hence propose to establish a second Pastry overlay among relay nodes that further distribute bandwidth stress. Instead of connecting to the server node directly, relay nodes self-organize into a second level of overlay and route to the server node through the Pastry routing scheme. Along the process, a Scribe multicast tree can be built on-the-fly to distribute the bandwidth stress accumulated at the server node. In addition, the second overlay can take advantage of the build-in Pastry PNS technique to exploit the network proximity at a higher level. In other words, while clustered overlay allows peers to exploit network proximity within the local bin at the local link level, the second

level overlay allows relay nodes to exploit network proximity across the bins at the major link level, since those relay nodes are much more likely to be located near the network access points. The two-level overlay is conceptually illustrated in **Figure 4.6** as follows.

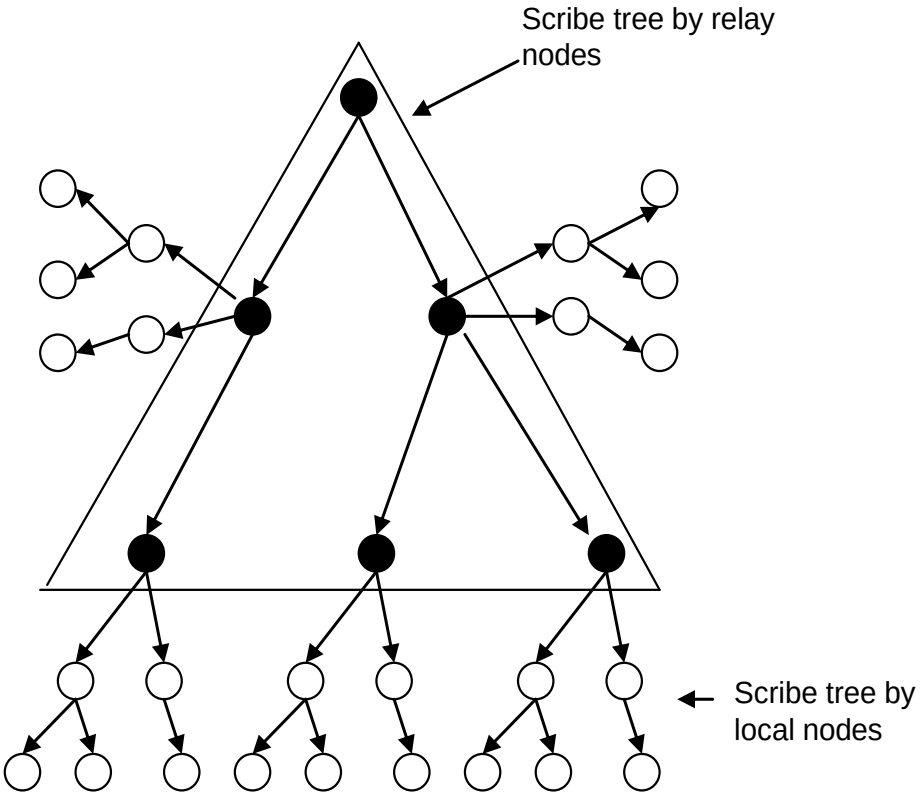
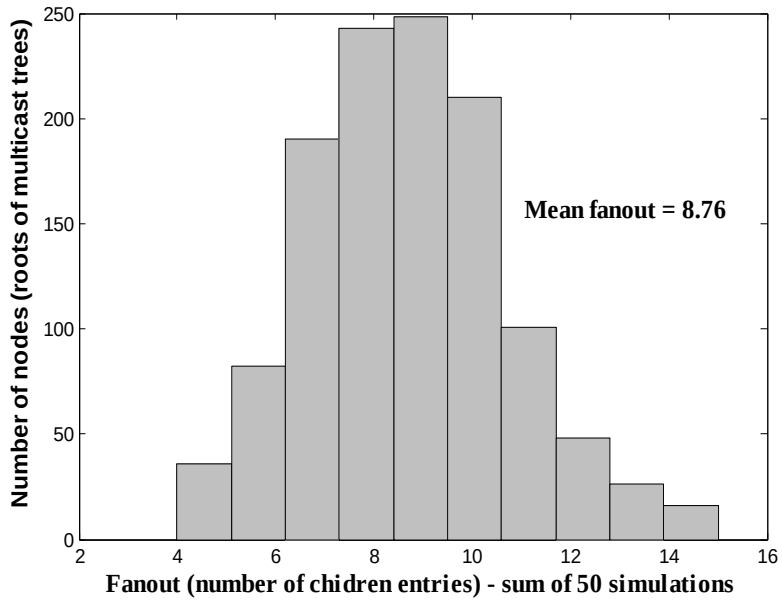


Figure 4. 6 Two Level Overlay

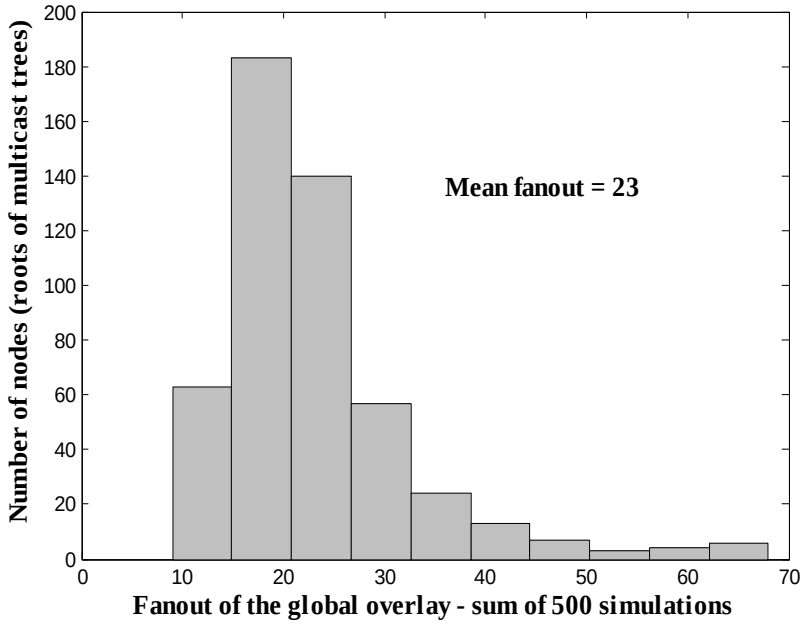
4.4.4.2 Simulation Procedure and Results

Following the establishment of 24-clustered overlays, we randomly select one end-host that is connected to a transit node in the topology as an AOI server node. The server node then pings a number of nodes in a local clustered overlay to select the closest node as the relay node for that local overlay. Note that the sever node only needs to know one node (e.g., through a central service) in any local overlay to be able to obtain the routing table of that node and has access to a list of nodes in that local overlay for pinging purpose. After the choice of relay nodes for each clustered overlay has been made, the relay nodes and the AOI server node then self-organize into a second level Pastry overlay. Given the relatively small number of the relay nodes (24 in our simulation case), we choose Pastry parameter b = 1 (binary ID for node), and leaf set size l = 4. When a peer joins the game session hosted by an AOI server node, it routes to the local relay node to join the local Scribe tree, and then the second level Scribe tree can be established when relay nodes route to the AOI server. The reverse path forwarding scheme is used to propagate state update traffic in the trees established (**Figure 4.6**).

To test the performance of the two level overlay, we experimented with a population level of an AOI of 240 players, which is a high population level for an AOI considering most current MMORPGs only support a handful of players in an AOI due to the bandwidth restriction. We have distributed those players uniformly among the local overlays. We measure the bandwidth stress of the AOI server node and relay nodes in supporting such a population level for those nodes being subject to most bandwidth stress as roots of the multicast trees. The results are shown in the figures as follows.



(a) Fan-out of AOI Server Nodes in the Two Level Overlay



(b) Fan-out of AOI Server Nodes in the Global Overlay

Figure 4. 7 Histogram of fan-out

Figure 4.7 (a) shows that the Two-Level Overlay has achieved an average fan-out of 8.76 at the roots of the multicast trees. For comparison purpose, we present the performance of a global overlay in terms of fan-out at the roots of the multicast trees for

supporting the same population level in **Figure 4.7** (b), which shows an average fan-out of 23. If each connection for a child needs an approximately 100kbps of upstream bandwidth stress to support the population level of 240 players in an AOI as discussed before, the Two-Level Overlay can effectively constrain the bandwidth stress of root nodes below 1Mbps (8.76 x 100kbps < 1000kbps), which is feasible for current high-end ADSL links. In terms of latency performance, **Figure 4.5** (a) shows that the CDF distribution of RDP of the Two-level Overlay is slightly worse than that of the 24 clustered overlay setting.

A comparison between **Figure 4.7** (a) and (b) further shows that, the bandwidth stress of AOI server nodes in the two-level overlay setting is symmetrically distributed around the centred average value, whereas those in the global overlay is positively skewed with the maximum value reaching 70. In both settings, the bottleneck removal algorithm can be used to reduce the stress of a congested node (Section 4.4.2).

In **Table 4.1**, we summarize the comparison between the four settings in supporting the population level of an AOI of 240 players: 1. the Global Overlay, 2. the Two-Level Overlay, 3. the 24-clustered overlay, and 4. the Central Server model.

Table 4.1 Comparison between four settings

	Global Overlay	**Two-Level overlay**	**24-clustered Overlay**	**Central Server Model**
Mean RDP	1.67	1.17	1.11	1
Expected Fan-out at roots (AOI servers)	23	8.76	24	240
Expected bandwidth stress for MMORPG traffic	2.3Mbps	876kbps	2.4Mbps	24 Mbps

Table 4.1 shows that, in supporting a population level of 240 players in an AOI, the cost to the central server would be comparatively high at 24Mbps to transmit the state update flow required by advanced MMORPGs such as World of WarCraft. A standard Scribe overlay can reduce the bandwidth cost by a factor of 10 through P2P multicast. However, this is at the cost of 67% increase of latency, which could significantly deteriotate the game responsiveness.

In comparison with the central server model, the 24 clustered overlay setting and the two-level overlay setting have improved the bandwidth distribution by factors of 10 and 27 respectively. The costs for both techniques are the increases of latency of 11% and 17% respectively, which are much less than the cost of the standard Scribe configuration (67%).

In comparison with the standard Scribe setting, both the settings of the two-level overlay and the 24 clustered overlays have improved the latency performance significantly up to 30 and 34 percent in mean value respectively. We believe such an improvement is significant with respect to the action responsiveness. For instance, highly real-time sensitive actions such as shooting require the latency to be less than 150ms [56]. If the IP routing distance from a player to the server is 100ms, the use of standard Scribe would increase the latency to 167ms that could have severe negative influence on the game playability. However, the use of our techniques only increases the latency to 111ms and 117ms, which might not be perceived by the users.

With respect to bandwidth distribution, the two-level overlay has outperformed the standard Scribe setting by a factor of 2.7. As a result, the two-level overlay is able to

support a high population level of 240 players in an AOI with practical peers' capacity such as ADSL links.

To conclude, we have demonstrated that our techniques can push the interaction metrics to a higher level, which would otherwise incur a higher cost using the standard Scribe setting with respect to latency, or an even worse cost for a central server with respect to bandwidth usage.

4.5 Related Work

Other popular structured P2P overlay designs include CAN [62], Chord [46] and Tapestry [63], each provides a scalable platform for P2P multicast. CAN differs from other designs mainly in that CAN uses a numerical distance metric to guide routing path through a Cartesian hyperspace, whereas Chord, Tapestry and Pastry all use hypercube routing algorithm. As a result, CAN multicast does not built multicast trees. Instead, it floods messages to all nodes in the CAN overlay through the routing tables maintained by the nodes. Multiple groups can be supported by creating a separate CAN overlay for each group, whereas Scribe builds a spanning tree for each group using a single Pastry overlay. Castro M. et al. [64] conducted a head-to-head comparison of P2P multicast between CAN and Scribe. The general findings are that, at the similar management and maintenance cost, Scribe outperforms CAN multicast on the order of 20% - 50% with respect to latency performance.

In comparison with Chord and Tapestry, the main advantage of Pastry is its relatively lower complexity. As a result, Scribe requires less management and maintenance overhead that benefits the scalability of P2P multicast. For instance, Bayeux [65] is built on top of Tapestry and supports multiple multicast groups. A root of a multicast group in Bayeux has to keep a list of all group members and all management traffic has to go through the root, whereas in Scribe the information for group member is distributed over the nodes.

While we have based our work on Scribe for reasons briefly discussed above, we believe that the techniques we developed can also be applied to CAN, Chord and Tapestry since they are all based on Distributed Hash Table.

The efforts to enhance the point-to-point routing performance of structured P2P overlay include Brocade [66] and Expressway [67]. Brocade separates "super-nodes" from normal nodes, and establishes a second overlay among the super-nodes. It is assumed that each node knows the AS domain that it belongs to (e.g., using BGP report). Each super-node maintains a "cover set" of a list of nodes in its AS domain and exchanges this information with other super-nodes for the selection of paths across domains. If a message's source and destination are across domains, the message is forwarded to the super-node for cross-domain routing. With respect to cross-domain routing, the super-node overlay approach is similar to our two-level overlay scheme in terms of exploiting network proximity at domain link level. However, in selecting the path across domains, our two-level overlay scheme takes advantage of the build-in PNS in Scribe, whereas Brocade relies on the exchange of state information of cover sets among super-nodes, which could have scalability concerns when the cover set involves a large number of nodes.

In response to the scalability concerns of Brocade, Expressway uses a summarization scheme to advertise the cover set. The summarization scheme is tailored for CAN that maps keys to Cartesian hyperspace. A hyperspace is partitioned into grids, and nodes residing in a common grid are summarized by the grid number. A super-node advertises the grid number instead of every node in the grid.

With respect to improving the point-to-point routing performance, the major advantage of our technique in comparison with Barocade and Expresseway is its light-weight nature by only applying PNS at the final hop of Scribe traffic path, whereas both Brocade and Expressway need to run different protocols and algorithms to maintain the super-overlay, reducing the scalability. Furthermore, the use of the second overlay in Barocade and Expressway is solely for the purpose of enhancing the point-to- point latency performance whereas our work has used the second overlay to distribute bandwidth stress required by traffic intensive applications such as MMORPGs, in addition to enhancing latency performance.

To our knowledge, we are the first to try to enhance the P2P multicast performance of Scribe in the context of MMORPGs.

4.6 Conclusions

In this chapter, we have developed a couple of techniques to enhance the P2P multicast performance of Scribe with respects to latency performance and bandwidth distribution.

The key to the first technique is to identify that the final hop of Scribe traffic path is topologically randomized and incurs the largest latency. Our proposed technique leverages clustering techniques and applies Proximity Neighbour Selection (PNS) to the final hop of the multicast traffic path. PNS is an effective light-weight technique that has been used by Scribe in controlling the distance of initial hops of its traffic path. It should be noted that the initial hops of Scribe traffic path usually involve links within local domains, and the final hop usually involves links across domains. Hence, the application of PNS to the final hop would be more effective in reducing the overall distance of a traffic path by exploiting network proximity at cross-domain link level. Simulation results have shown that the technique can improve the latency performance by more than 30 percent in comparison with what can be achieved in a standard Scribe multicast.

The second technique builds a two-level overlay: the higher level among nodes located close to network access points, the lower level among nodes in a local domain. While PNS can be applied at both levels for latency performance, the bandwidth stress required by applications can now be distributed among the nodes in the higher level overlay. The simulation results have shown that, while latency performance is slightly worse than what is achieved by the first technique, the second technique has improved bandwidth distribution by up to 2.7 times, in comparison with what can be achieved by a standard Scribe overlay.

We have developed these techniques in the context of Massively Multiplayer Online Games (MMORPGs). The current central server model in supporting MMORPGs faces critical scalability challenges. Game distributed design may leverage Scribe to achieve a

scalable deployment of MMORPGs. The fundamental attractiveness of MMORPGs lies in the real-time interactions between players. From the technical perspective, the quality of interactions can be translated into two metrics. The first is the population level of an AOI, which is defined as the number of players who can interact with each other simultaneously. The second is the responsiveness of the game to player actions, which is defined as the time difference between when an action is issued by a player and when the action is executed and recognized by other players in the game world. Both of the interaction metrics are directly constrained by the bandwidth usage of the game traffic and network latency respectively.

The techniques we proposed could benefit the distributed game design over the structured P2P system in improving the interaction metrics for better game playability.

Chapter 5. Supporting a Seamless Map in Peer-to-Peer Systems for MMORPGs

Abstract

Massively Multiplayer Online Role Playing Games feature huge maps that can be partitioned into smaller components called locales to achieve scalability. In peer-to-peer (P2P) systems, those locales can be assigned to peers who are willing to take up the role of locale servers. A seamless map requires inter-locale communications to allow interactions between players across the locale boundaries. In assigning locales to peers, it is critical to seek approaches to reduce the inter-locale latency cost in a P2P system. In this chapter, we formulate the Locale Assignment Problem for P2P system and demonstrate that the problem is NP-hard. We hence propose a low-cost heuristic that partitions the physical network into bins, as well as, aggregating the partitions of the virtual world by clustering neighbouring locales in a game map. We then demonstrate that by assigning clustered neighbouring locales in the virtual map into the peers in common physical network bins, we are able to achieve inter-locale latency cost for the Locale Assignment Problem that is close to the output of the optimization model we formulated, while avoiding the high computation cost required by the latter.

5.1 Introduction

The general approach for games to achieve scalability is to partition the virtual world into smaller components called locales. In a Central Server model, the simulation of these locales can be distributed among the servers in a server cluster, whereas in a P2P environment, these locales can be assigned to the peers who are willing to take up the role of locale servers.

If the locales of a game virtual map are segregated from each other, then there would be no communication between locale servers. In this case, certain techniques are required to give the player the impression that the virtual world is contiguous. For instance, when a player moves from one locale to another, he or she has to walk through a path or fly across a valley to be disconnected from one locale server and connected to another. In Chapter 3, we have conducted a traffic trace at the client side for World of Warcraft. The traffic trace recorded in **Figure 3.4** presents some evidence for the application of this technique. For instance, as shown in **Figure 3.4**, when the player travels from one part of the game world to another, the transition of the locale servers can be identified by a peak load of bandwidth usage for the downloading of locale specific objects. In addition, when the player flies thorough the valley, there is a clear drop off of the bandwidth usage. We believe that behind the scene, the server is processing the transition and the time taken for flying through the valley has served to give the player the impression that the transition is seamless.

While these techniques can obviously ease the management of inter-server communications, they also impose constraints on game design, particularly on the scalability of the locales. For instance, Pittman et al. [68] conducted a population study of World of Warcraft and found that the distribution of the player population in the game virtual world is non-uniform, and the relationship between the player population and the locales generally follows a power-law distribution. While 75% of the locales have only a small number of players, 5% of the locales have more than 40 players. Furthermore, several most popular locales in the game world have more than 100 players. Those popular locales are mainly places in the virtual world, such as market places, that encourage player congregation for social activity. Given the approach of

using segregated locales for the game world, the states of each locale are usually managed by a single server. Hence, the scalability of the locale is largely constrained by the capacity of the server. As a result, when a locale is highly populated, further connection requests may be rejected, which would negatively affect player experiences.

On the other hand, to create a seamless map with transparent boundary between locales, servers of neighbouring locales need to exchange state information of players to enable the cross-boundary interactions between players. We envisage that a map of seamless design would have significant potential to enhance game features for better playability. For instance, a widespread battlefield that engages a large number of players or a social congregation of hundreds of participants, could adopt a seamless map that gives the players the impression of a contiguous virtual space without boundary limitations. It should be noted that while the scalability of a locale can be enhanced by joining neighbouring locales into a seamless map, a cost would be incurred by the inter-locale communication between neighbouring locale servers, which is the focus of this chapter.

The cost of inter-locale communication might be of less concern for servers in a central server cluster since the traffic is mainly carried by the Local Area Network (LAN). However, in a P2P system, the communication cost could be high if the neighbouring locales are assigned to peers with limited bandwidth and far apart from each other in terms of latency. Hence, it is important to find an effective approach to assign locales to peers to reduce the inter-locale communication cost. Hereafter we refer to the problem of finding such an approach as the '*Locale Assignment Problem'*.

Our works presented in Chapter 3 and Chapter 4 have been mainly on the design of P2P system to support MMORPGs. In Chapter 3 and Chapter 4, our model assumed a game world partitioned into segregated locales. In this Chapter, we focus on the support of a seamless map and consider inter-locale communication cost that leads to the Locale Assignment Problem. This is the natural continuation of our effort to build a framework for supporting MMORPGs using a P2P system .

To this end, we propose a heuristic to partition both the physical network and the virtual world (game map) to control the communication cost. In partitioning the physical network, we are actually trying to group peers that are topologically close to

each other and select them to be potential locale servers. We then demonstrate that the Locale Assignment Problem is NP-hard and incurs a high computational cost even restricting the assignment of locales to a small set of peers selected through the physical network partition. As a result, the proposed heuristic aggregates the locales in a virtual map into larger partitions. In partitioning both the physical network and virtual game map, we are able to assign a group of neighbouring locales to a group of peers who are topologically close to each other. Through the heuristic, we demonstrate that we are able to achieve a near optimal inter-locale communication cost when deploying a seamless map in a P2P system.

The rest of the chapter is organized as follows. In section 5.2, we first discuss the rationale and approach for physical network partition. Then through formulation, we demonstrate the NP-hardness of the Locale Assignment Problem. We hence propose to further partition the virtual world. We finally present the dual-partition heuristic. In section 5.3, we present simulation procedures and results. We finally discuss related work in section 5.4 and conclude in section 5.5.

5.2 Dual-Partition Heuristic

In this chapter, we propose a dual-partition heuristic that partitions both the physical network and the virtual map for supporting a seamless map in a P2P system. In partitioning the physical world, we are actually exploring the physical network proximity in order to select potential peers as candidate locale servers with coarse granularity. In partitioning the virtual world, we aim to optimize the inter-locale communication cost by exploring proximity of virtual locales in a seamless map.

5.2.1 Physical Network Partition

Clustering technique is widely used in the research literature [55, 58, 59]. The fundamental idea is to find nodes that are close to each other, and provide this locality information to applications to achieve better performance (see Section 4.4.3). Simulation through a Transit-Stub [60] topology has shown that the RTT between nodes

within the same bin is on average 4.06 times shorter than the RTT between nodes across bins.

After clustering nodes into bins, instead of assigning game locales to actual hosts, we assign locales to bins. In a large-scale P2P system, the number of available peers who are qualified to be the locale servers could be high. As demonstrated in later sections, the Locale Assignment Problem incurs high computation cost. By assigning locales to bins instead of actual hosts, we are able to significantly reduce the search space. For instance, instead of trying to find an optimal assignment scheme among hundreds of available peers, we cluster those peers into a small number of bins. An effective clustering scheme ensures that assigning locales to different hosts in a common bin would incur similar inter-locale communication cost.

Furthermore, the issue of churn [69] is a critical concern for any P2P system design. Churn refers to the phenomena that in a P2P system, peers or client computers come online and go offline unpredictably. Hence, any optimization scheme or heuristic for the Locale Assignment Problem that seek reduced inter-locale communication cost by finding the optimal assignment of game locales to peers would be impractical. This is particularly true for MMORPG given the high churn rate of peers [70] in online games. However, by assigning locales to bins instead of particular hosts, if churn occurs, for instance, a locale server goes off-line, another backup peer in the same bin can take up the role of the locale server without affecting much the inter-locale communication cost.

5.2.2 Formulation

After clustering nodes into bins, we formulate the Locale Assignment Problem as follows.

The known variables are:

d_{st} : The mean delay between bin s and bin t.

m_{ij} : The number of avatars in the boundary of locale i and locale j.

c_{ij}^{st} : The delay cost of state information exchange between locale i and locale j if assigning locale i and locale j to bin s and bin t respectively. This is defined below:

$$c_{ij}^{st} = m_{ij} \text{ x } d_{st} \qquad (1)$$

N: The number of locales in a seamless map.

M: The number of physical bins.

b_s: The number of peers who can become locale servers in bin s, $1 \leq s \leq M$.

The decision variable is

$$x_i^s = \begin{cases} 1 \text{ if zone i is assigned to bin s} \\ 0 \text{ otherwise} \end{cases} \qquad (2)$$

The objective function is

$$\text{Minimize} \sum_{\forall i,j} \sum_{\forall s,t} c_{ij}^{st} x_i^s x_j^t \quad i \neq j \qquad (3)$$

subject to

$$\sum_{s=1}^{N} x_i^s = 1 \quad \forall \text{ i: } 1 \leq i \leq N \qquad (4)$$

$$\sum_{i=1}^{M} x_i^s \le b_s \ \forall \text{s: } 1 \le \text{s} \le M \qquad (5)$$

To transform (3) into a linear programming problem, we introduce a binary variable y_{ij}^{st} such that

$$y_{ij}^{st} = \begin{cases} 1 \text{ if } x_i^s = x_j^t = 1 \\ 0 \text{ otherwise} \end{cases} \qquad (6)$$

The objective function (3) then becomes

$$\text{Minimize } (\sum_{\forall i,j} \sum_{\forall s,t} c_{ij}^{st} y_{ij}^{st}) \text{ i} \neq \text{j} \qquad (7)$$

subject to

$$\sum_{s=1}^{N} x_i^s = 1 \quad \forall \text{ i: } 1 \le \text{i} \le N \qquad (8)$$

$$\sum_{i=1}^{M} x_i^s \le b_s \ \forall \text{s: } 1 \le \text{s} \le M \qquad (9)$$

$$y_{ij}^{st} \le x_i^s \qquad (10)$$

$$y_{ij}^{st} \le x_j^t \qquad (11)$$

$$x_i^s + x_j^t \le 1 + y_{ij}^{st} \qquad (12)$$

The constraints in equation (8) ensure that each locale in the map can only be assigned to one bin. The constraints in equation (9) ensure that the number of locales assigned to a bin is less than or equal to the number of peers in that bin who are willing to take up the role of the servers for the locales. The constraints in (10) (11) and (12) are to make sure that equation (6) is satisfied.

5.2.3 NP-Hardness of the Locale Assignment Problem

Given the objective function in (3), we assume t is a constant T. We further assume $x_j^t = x_j^T = 1$. In other words, for each pair of neighbouring locales, one has to be assigned to a particular bin T. An example is given in Figure 5.1 to demonstrate the feasibility of such an assignment scheme.

Then (3) becomes

Minimize $\sum_{\forall i,j} \sum_{\forall s} c_{ij}^{sT} x_i^s$ (13)

subject to (4) and (5).

This model has been proved to be equivalent to the well-known Bin Packing Problem[71], which is NP-hard. Hence, the Locale Assignment Problem is NP-hard.

N	T	N	T
T	N	T	N
N	T	N	T
T	N	T	N

Figure 5.1 T – locales assigned to bin T, N- locales assigned to any bin but T.

5.2.4 Virtual World Partition

The computational cost could be prohibitively high for Locale Assignment Problem, even if we choose to assign locales to a limited number of bins using the model formulated above.

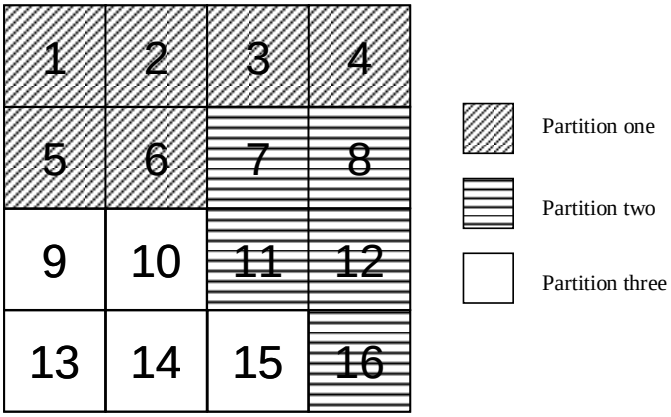


Figure 5.2 Partitioned map

For an equally divided square map as shown in **Figure 5.2**, each locale has at least two neighbouring locales. Let the number of locales be N, there are at least 2N pair of neighbouring locales. Let M be the number of network bins partitioned, there will be M(M-1) possibilities to assign each pair of neighbouring locales to the bins. Hence, the number of y_{ij}^{st} decision variables is at least NM(M-1) or $O(NM^2)$. The number of constraints is N + M + 2NM(M-1) or $O(NM^2)$. Given the complexity of the model, as demonstrated in section 5.3, the formulation can only solve the Locale Assignment Problem in real time for a seamless map having a limited number of locales.

As a result, in addition to physical network partition as discussed previously, we propose to judiciously aggregate the locales in a virtual world map into large partitions to reduce the cost of the Locale Assignment Problem. While a seamless map has multiple locales, neighbouring locales can be grouped together and regarded as a single partition to be assigned to a bin. For instance, if a seamless map has sixteen locales as demonstrated in **Figure 5.2**, neighbouring locales 1 – 6 could be grouped together to form a partition and assigned to peers in a common bin.

The locale grouping policy is a design issue that should consider performance requirement of a seamless map. The locales grouped in one partition, when assigned to peers in a common bin, would have better capacity for cross-boundary interactions between players since the inter-locale latency incurred between peers in a common bin would be low. In other words, as the example shown in **Figure 5.2**, the map design for locales in partition one could allow more cross boundary interactions. However, since locales in partition two are assigned to another bin, interactions across locale 3 and 7, for instance, should be kept low since the inter-locale latency could be high. As an example, the map for partition one could be designed to handle crowded spaces like cities, battlefield and markets. On the other hand, for boundary area between locales in different partitions, such as locale 3 and 7, the map should be designed as walls or paths to avoid excessive inter-locale latency.

5.2.5 Selection of Bins

In selecting candidate bins for assigning the locales, we consider the delay cost incurred by the latency between player hosts to the locale servers. To that end, we sort bins based on the sum of delay cost S_t. Let P_s be the number of peers in bin *s* and d_{st} be the delay between bin *s* and *t*. S_t is defined as:

$$S_t = \sum_{\forall s} P_s d_{st} \qquad (14)$$

Hence, S_t represents the sum of latency cost from player hosts to locale servers if all locales are assigned to bin *t*.

Based on discussions so far, we present the dual-partition heuristic in **Figure 5.3**. In the following sections, we will compare the results obtained from running the optimization model and the heuristic. When presenting the results of the heuristic, we will discuss the heuristic in more detail.

1. Partition the physical network into *M* bins.

2. Sort the physical network bins based on s_t such that

$$s_1 < s_2 < \cdots < s_M$$

3. Let b_i be the number of available zone servers in bin i.

4. Let N be the number of zones in a map.

5. Select the first n bins such that

$$\sum_{i=1}^{n} b_i \geq N$$

6. When i = n, let $b_n = b_n^{'}$ such that $\sum_{i=1}^{n} b_i = N$

7. Sort the selected bins such that

$$b_1 \geq b_2 \geq \cdots \geq b_n$$

8. for i = 1 to n
9. if feasible
10. group b_i neighboring zones along the edge of the map
11. else
12. group b_i neighboring zone
13. end

Figure 5.3 Dual partition heuristic

5.3 Simulation Procedures and Results

5.3.1 Topology Setup

We used Georgia Tech. GT-ITM to generate a transit-stub (TS) topology for experimental purposes. The topology have nodes grouped into either transit or stub domain. In our topology, there are three transit domains with on average eight nodes (transit nodes) in each domain. Each transit node is connected to three lower level stub domains on average. There are on average eight nodes (stub nodes) in each stub domain. TS topology emphasizes the locality feature of the Internet. To this end, link weight is assigned such that traffic path between two nodes in the same domain will remain within that domain [60]. In [61], a similar topology is used to model the Internet. We then have 15 hosts on average connected to each node in the topology to model an overlay network of 10000 hosts. We use the topology assigned link weight as RTT for the links. The link between an end-host and a topology node is assigned a random RTT value between 3 – 10 ms.

We then take advantage of the locality properties of the TS topology by clustering nodes based on the domain they belong to. Thus, we are able to have 24 clusters of nodes because there are 24 stub domains in our topology model. We consider each clustered stub domains as a candidate bin for the Locale Assignment Problem. Given our topology, we have on average 420 end-hosts (peers) in each bin.

5.3.2 Results of Solving the Optimization Model

To instantiate the optimization model formulated in Section 5.2.2, we randomly select a small number of peers as locale servers. In **Table 5.1**, we list the number of available locale servers in the first five bins sorted based on formula (14).

Table 5.1 Available locale servers in bins

Bin Index	1	2	3	4	5
Number of available locale servers	3	7	4	5	12

It should be noted that in **Table 5.1**, the bins are sorted based on the value of S_t as defined (14) such that, for instance, the model should try to assign locales to bin one first, since the sum of delay cost incurred by assigning locales to bin one would be minimal.

We choose Cplex optimization software to optimize the model formulated in Section 5.2.2 on a Pentium IV 2Ghz Linux server with 2 GB of RAM. We experimented with a square map and gradually increased the number of locales from 12 to 25. The results are presented in Figure 5.4. The time taken in solving the optimization model increases as the number of locales increases, and is listed in **Table 5.2**. The results of running the optimisation model are shown in Figure 5.4.

Table 5.2 Time taken to run the optimization model

Number of locales	12	16	20	25
Time taken to run the optimization model (seconds)	0.22	22	1386	34709

2	2	3	3
2	2	1	1
2	2	2	1

(a) 12 zones

3	3	2	2
3	3	2	2
4	1	2	2
4	1	1	2

(b) 16 zones

5	2	1	3	3
2	2	1	3	3
2	2	1	4	4
2	2	4	4	4

(c) 20 zones

2	2	2	4	4
2	2	1	4	4
2	2	1	1	4
5	5	5	3	3
5	5	5	3	3

(d) 25 zones

Figure 5.4 Results of optimization model

In **Figure 5.4**, each partitioned square represents a locale in the map. The number in the square represents the bin number that a corresponding locale is assigned to. For instance, in **Figure 5.4** (a), three locales are assigned to bin one, seven locales are assigned to bin two and the remaining two locales are assigned to bin three.

From **Figure 5.4**, it can be first observed that, the optimization model assigns locales to bins in a sorted order as expected. For instance, there are three peers available to be locale servers in bin one, the optimisation model populates those peers first before continuing to populate bin 2, 3, 4 and 5. In doing so, the model ensures the sum of the latency cost between players and locale servers is minimal as discussed in the previous section.

Secondly, the optimisation model groups neighbouring locales in the virtual map and assigns them to the common bin to achieve minimal inter-locale latency cost. It can be observed in the figure that, locales assigned to a common bin are placed together, with best efforts, in a continued fashion without being separated by other locales. This is because the inter-locale latency cost is incurred by interactions across shared boundaries of neighbouring locales. Grouping neighbouring locales together and assigning them to the common bin would be beneficial in reducing the inter-locale latency cost since the inter-locale traffic is constrained within the peers in a common bin, in which peers are close to each other in terms of latency.

Finally, the optimization model is not scalable in terms of computation cost. For instance, as shown in **Table 5.2**, when the number of bins is above 16, the computation cost will become prohibitively high in solving the model in real time.

5.3.3 Results of Running the Heuristic

Enlightened by the results in **Figure 5.4**, our general intuition for the heuristic is that if we partition the virtual map and cluster the neighbouring locales, and then assign the entire partition to a common bin, the inter-locale latency cost should be comparable to the results achieved by the optimization model.

We experimented with a 25 locale map. As shown in **Figure 5.5**, we applied the heuristic in **Figure 5.3** to partition the map by clustering neighbouring locales based on available locale servers in each bin (**Table 5.1**).

For instance, in bin one, there are three available peers who are willing to take up the roles as locale servers, we hence cluster three locales to be assigned to bin one as a single partition. When we cluster locales in the virtual map, we try to place the big partition, for instance, partition for bin two in which there seven peers who are willing to take up the roles as locale servers, along the edge of the map to reduce the number of shared boundaries between different partitions (step 6-13 of the heuristic). An example is given in **Figure 5.6**.

3	3	2	2	2
3	3	2	2	2
4	1	1	1	2
4	4	5	5	5
4	4	5	5	5

Figure 5.5 25 Locales - Partitioned Map

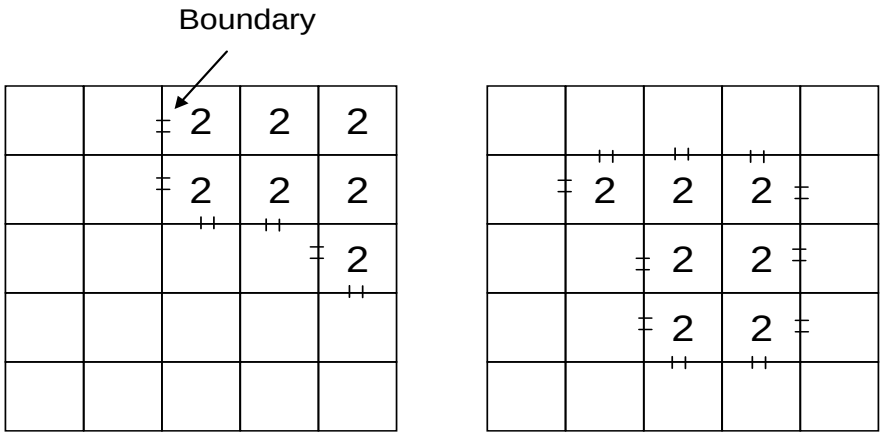


Figure 5.6 Boundary Comparison

If we have partition two (all locales in partition two are assigned to peers in bin two) placed along the edge of the map, we will have six boundaries shared with other partitions as shown on the left part of **Figure 5.6**. However, if we have the partition placed in the middle of the map, we would have eleven boundaries as demonstrated in the right part of the figure. Since inter-locale communication is incurred by cross-boundary traffic, we would expect less latency cost if we had less shared boundaries when we assign the partition along the edge of the map.

To compare the performance in terms of the inter-locale latency cost between **Figure 5.4** (d), which is the result of solving the optimisation model, and **Figure 5.5**, which is achieved by running the heuristic, we assign locales of the map to randomly selected peers in the corresponding bin. For instance, based on **Figure 5.5**, we assign the four locales on the top left corner of the map to four selected peers in bin three, whereas based on **Figure 5.4** (d), we assign the four locales on the bottom right corner to four selected peers in bin three. It should be noted that when assigning locales to peers in the same bin, we do not further seek optimization goals because of the issue of churn in P2P system as we discussed in Section 5.2.1.

To normalize the simulation results, we set $m_{ij} = 1$ (section 5.2.2). Again, we use the network topology as discussed in Section 5.3.1. We run the simulation for 10000 times and the results are demonstrated in **Figure 5.7**.

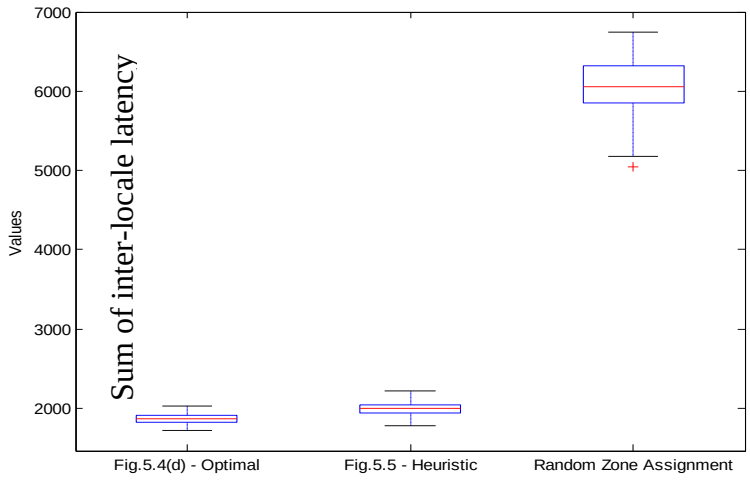


Figure 5.7 Comparison of the sum of the inter-locale latency

Box plots in **Figure 5.7** shows that the sum of the inter-locale latency based on **Figure 5.5** is only slightly higher than that of **Figure 5.4**(d), which is obtained by solving the optimization model. The mean of the sum of the latency of the optimized model (**Figure 5.4**(d)) is 1847, whereas the mean of the sum of the communication cost of the partitioned map (**Figure 5.5**) is 1915 and only 5 percent higher than that of the optimized model.

Given our particular topology, m_{ij} = 1 (one player at each boundary of each locale) and a 25-locale map has 45 shared boundaries altogether, the player at the boundary of a locale would be subject to an extra 41ms delay cost on average due the requirement of the inter-locale communication for the optimized model (1847/45 = 41), whereas for the partitioned map achieved through the heuristics, the player would be subject to 42.6 ms delay cost (1915/45 = 42.6). The minor difference of 1.6 ms in delay cost is negligible as far as the game quality of MMORPG is concerned.

For comparison purpose, we finally assign the locales of a 25 locale map to the peers in the topology randomly and calculate the sum of the inter-locale latency. As shown in **Figure 5.7**, both the optimized model and the partitioned map can save the communication cost by a factor of three in comparison with a map for which locales are randomly assigned to the peers.

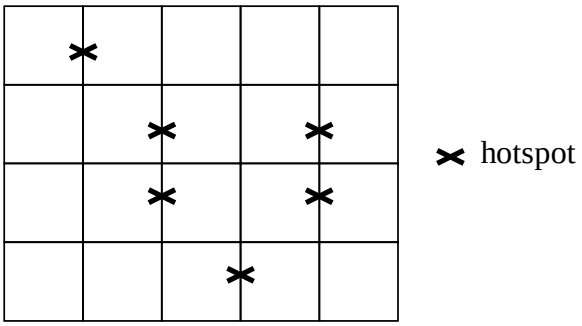
To conclude for this section, the computation cost for the Locale Assignment Problem is prohibitively high (**Table 5.2**) when solving the optimization model. We are able to achieve nearly the same level of optimization goal in reducing the inter-locale latency by partitioning the locales in the virtual map and assigning the partitions to physical network bins.

5.3.4 Map with Hotspots

In the previous simulation, we have set $m_{ij} = 1$ across all boundaries. In other words, players are uniformly distributed across the boundaries of the map. However, in MMORGP, player flocking may occur due to the features of the game design. For instance, an interesting occurrence of an event may attract more players to come to a certain part of the game world and create a hotspot in the map. To experiment with a

map with hotspots at the boundaries, we selected a number of boundaries of a 20-locale map and set $m_{ij} = 2$ at those boundaries to simulate hotspots at which the interactions between players are twice as much as that of a normal boundary. In **Figure 5.8**(a), those hotspots are marked with crosses.

We then run the optimization model for the Locale Assignment Problem and the results are presented in **Figure 5.8**(b). It can be observed from the figure that, in addition to what we have discussed in Section 5.3.2, the optimization model restricts the hotspots within common physical network bins. In other words, the optimization model assigns the two neighbouring locales connecting a hotspot to a common bin. In doing so, the communication cost incurred by the hotspot would be kept low since the two neighbouring locales are assigned to peers in a common bin, and hence are close to each other in terms of latency.



(a) Map with hotspots

3	3	2	2	2
3	2	2	2	2
3	1	1	4	4
5	1	4	4	4

(b) Results of Optimization Model

3	3	1	2	2
3	1	1	2	2
3	4	4	2	2
5	4	4	4	2

(c) Partitioned Map

Figure 5.8 A Map with hotspots at boundaries

Enlightened by the results we discussed above, we again apply the virtual map partition following the heuristics in **Figure 5.3**. In addition, while assigning the big partitions along the edge of the map, we restrict the neighbouring locales connecting a hotspot to be assigned to peers within a common physical network bin. The partitioned map following the heuristics is presented in **Figure 5.8** (c).

The total communication costs of both **Figure 5.8** (b) and **Figure 5.8** (c) are compared in **Figure 5.9**, following the procedures similar to what has been described in the previous section for comparing the optimisation model and the heuristics.

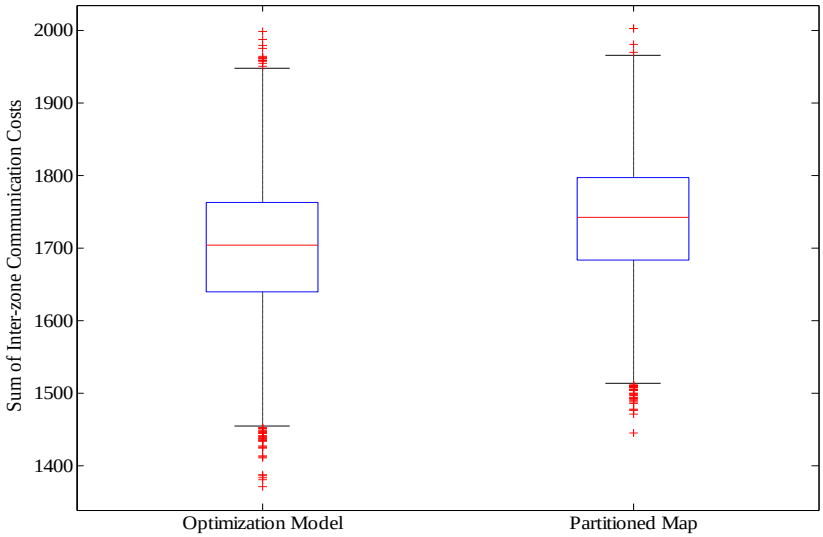


Figure 5.9 Comparisons of the sum of latency

In **Figure 5.9**, the mean of the sum of the latency cost of the optimization model (**Figure 5.8**(b)) is 1699, whereas the mean of the sum of the latency cost as a result of running the heuristic (**Figure 5.8**(c)) is 1739, less than 3 percent higher than that of the optimized model.

Given that a 20-locale map has 28 shared boundaries in total, the player at the boundary of a locale would be subject to an extra 60.7ms delay cost on average due the requirement of the inter-locale communication for the optimized model (1699/28 = 60.7), whereas for the partitioned map as a result of the heuristics, the player would be

subject to 62.1 ms delay cost (1739/28 = 62.1). The minor difference of 1.4 ms in delay cost is negligible as far as the game quality of MMORPG is concerned.

To conclude for this section, if games feature hotspots at certain area of the map that could lead to increased inter-locale latency costs at the boundaries, the map that is partitioned based on the location of the hotspots could achieve nearly the same level of latency costs as that can be achieved by the optimization model. We again demonstrated the effectiveness of partitioning both the physical network and virtual map in reducing inter-locale latency costs, while avoiding the high computation costs required by the optimization model.

5.4 Related Work

To our knowledge, there is no existing work that directly addresses seamless maps design in P2P system. Distributed server architecture is usually considered as an alternative to central server model in addressing the scalability and performance bottleneck of the latter. In [71] and [72], heuristics are developed for distributed server system to improve performance. The main focus of those heuristics is to assign players to servers so that the communication cost between the players and the servers (e.g., latency) is low, whereas our focus is to assign locales to servers so that the communication costs in terms of latency between servers is low.

In [72], inter-server communication cost is addressed as part of a design of partitioning scheme for distributed server system. However, the heuristic developed is subject only to a small number of servers (e.g. 8 servers). Given the partitioning of physical network and virtual map, our approach is not limited by the number of servers. This is important since in P2P system, a virtual map might need to be partitioned into much smaller components to be assigned to peers with smaller capacity comparing to servers in a distributed server system. As a result, more peers are required for the Locale Assignment Problem.

It should also be noted that distributed server system is relatively static, whereas P2P system features dynamics of peer participation or churn. As a result, we choose to assign locales to bins instead of particular hosts.

5.5 Conclusions

Massively Multiplayer Online Role Playing Games (MMORPGs) usually feature a huge virtual world for players to explore. To achieve deployment scalability, the game world can be partitioned into smaller components called locales. In a peer-to-peer system, those locales can be assigned to peers who are willing to take up the roles of locale servers to achieve scalability. A seamless map requires inter-locale communications to allow interactions across locale boundaries between players. A seamless map improves game playability by making the boundaries of locales transparent to players. To enable a seamless map in P2P system, it is critical to find effective approaches such that when assigning locales to peers, the inter-locale latency cost can be kept minimum. In this paper, we formulated an optimization model for the Locale Assignment Problem and demonstrated the NP-hardness of the problem. We then proposed a low-cost heuristic to partition the physical network into bins such that peers in a common bin are topologically close to each other. Furthermore, enlightened by the results of the optimization model, the heuristic partitions the virtual world map such that neighbouring locales in the map are clustered together and assigned to a bin as a single partition. In doing so, we demonstrate that we are able to achieve the same level of inter-locale latency costs as those can be achieved by the optimization model we formulated, while avoiding the prohibitively high computation cost required by the latter.

Chapter 6. The Study of Churn Impact on Structured Peer-to-Peer Overlay in Supporting MMORPGs

Abstract

In this Chapter, we conduct a performance study on a distributed game design that supports MMORPGs through structured P2P overlay, focusing on the churn aspect – the continuous process of node arrival and departure. Through discrete event simulation, we show that the distributed game design should consider a trade-off between the real-time stringency of the game features and the churn resilience. For instance, in the distributed game design, Map Type 1 represents a battlefield in the game world that emphasizes real-time stringent quick actions such as shooting, hence less resilient to churn; whereas Map Type 4 represents a market city in the game world that emphasizes a large group size of concurrent players at the cost of relaxed real-time stringency for actions, hence more resilient to churn. Our simulation shows that the churn resilience of Map Type 1 is nearly 30% better than that of Map Type 4. Furthermore, to address the performance concerns of a global structured P2P overlay, we propose a clustered overlay architecture by grouping nearby nodes into clusters. We show that the clustered overlay achieves an overall performance gain with respect to latency and churn, without incurring extra overlay maintenance cost. Finally, we show that the structured P2P system is promising in supporting MMORPGs, from the perspective of churn resilience. With a pre-defined quality-of-service, the simulation has shown that over 95% game sessions can achieve the quality-of-service.

6.1 Introduction

From the perspective of network resources, the major concerns for a P2P overlay in supporting MMORPGs are *performance* and *reliability*. Our works in Chapter 3, 4 and 5 have mainly focused on the performance aspect: in Chapter 3 and 4, we have addressed the bandwidth distribution and latency optimization of a P2P overlay to fulfil the requirement of MMORPGs. In Chapter 5, we proposed a heuristic to optimize the inter-locale communication in the game world to facilitate a seamless map design. In this Chapter, we focus on the reliability aspect to further establish the feasibility of a P2P overlay in supporting MMORPGs.

In a P2P overlay, a key contributor to reliability is how the system performs in response to churn – the continuous process of peer arrival and departure. Since peers contribute bandwidth and other resources to the system, the churn of a peer may impact the quality of the game for other peers. For instance, in Chapter 3 and 4 we proposed to use a multicast tree to multicast the game traffic. In a multicast tree, a peer may be assigned the duty of being a relay node by the overlay protocol to multicast game traffic to downstream nodes. It should be noted that the relay node is also a participant in the game. Assuming the selfish-nature of peers, a relay node may choose to leave the overlay after his/her session duration in the game world is completed. As a result, the downstream nodes will be dropped off the multicast tree at the time instant the relay node leaves the overlay. While a well designed P2P overlay protocol will enable the dropped peers to reconnect to the tree, during the reconnection process, the dropped peers will miss the game traffic that may reduce the game quality. From the perspective of a player, the churn causes an interruption to an otherwise continuous game session.

In this chapter, we continue our effort in the study of Scribe, a multicast tree that is built on top a structured P2P overlay in supporting MMORPGs, focusing on the churn impact perspective. In comparison to a Central Server environment in which the central service is always available, the interrupts caused by churn are unique to a P2P environment. As a result, the study of churn and the interrupts caused by churn is of paramount importance in establishing the feasibility of a P2P overlay in supporting MMORPGs.

The rest of the chapter is organized as follows. In Section 2, we briefly review our previous works related to a game distributed design that supports MMORPGs through structured P2P overlay, since the study of churn is closely related to the distributed game design. In Section 3, we present a static analysis of the property of the P2P overlay in relation to churn. In Section 4, we present a discreet event simulation that analyses the churn impact on the structured P2P overlay in supporting the distributed game design quantitatively. We discuss related works in Section 5, and conclude in Section 6.

6.2 A Review of the Distributed Game Design over Structured P2P Overlay

In this section, we briefly review our previous works in Chapter 3 and Chapter 4 on a distributed game design that support MMORPG through a Pastry overlay, a typical structured P2P design. We emphasize a few key points that are of relevant to the churn performance study.

6.2.1 Game World Partition

Current MMORPGs usually feature huge virtual worlds consisting of battle fields, training schools, cities, and markets, in which a player can explore through the control of an avatar – a representation of a role character in the virtual world.

The general approach for games to achieve scalability is to partition the virtual world into smaller areas called locales. In a Central Server model, the simulation of those locales can be distributed among the servers in a server cluster, whereas in a P2P system, those locales can be distributed among the peers who are willing to take up the roles of locale servers.

In our study, each locale constitutes an Area of Interest (AOI), in which players receive the same set of game state updates from the locale server. We assign the duty of locale servers to a small set of super peers (e.g., %4 of overlay population for churn study), which are peers with good computing power and network resources. To study the churn performance, we established a simulation model of Pastry overlay consisting of 10,000 peers, and had peers grouped into locales considering the configurations in current MMORPGs. Each locale corresponds to a Map Type in the game world that features a typical game environment available in current MMORPG. Hereafter, we will use the term map and locale interchangeably, and the meaning will be made clear in the context. The details of the game world partition into locales in our distributed game design are listed in **Table 6.1** as follows.

Table 6.1 Game world partition

	Map type 1	Map type 2	Map type 3	Map type 4
State update bandwidth requirement (kbps)	10	20	30	40
Interactivity	Player vs. Player fighting	Player vs. Non-player fighting	New player training school	Market city
Real-time sensitivity	High	Medium	Medium	Low
Number of maps (locales)	120	70	40	24
Number of players supported (group size)	20	40	60	100
Total number of players (group size)	2400	2800	2400	2400

It should be noted that, in comparison to the simulation setup in Chapter 3, we have only increased the overlay size from 5000 peers to 10,000 peers. In conducting the churn study, we intend to compare the churn impact between the global overlay and the clustered overlay models (see Chapter 4). The increased overlay size is to facilitate the application of the clustering technique. For instance, for a global overlay size of 10,000 peers, after clustering the global overlay into three clustered overlays, a clustered overlay will still have more than 3000 peers on average, which is a reasonable overlay size for scalability concerns, as well as, simulation purposes.

Our distributed game design applies a hybrid approach via the assignment of locale servers to super peers. Furthermore, the group size of each locale takes into consideration the interaction level and bandwidth requirement of the game traffic in the locale. These will be discussed briefly in the following sections. For detailed analysis and discussion, please refer to Chapter 3.

6.2.2 A Trade-off between Bandwidth and Latency

Based on the locale server assignment, the game traffic model in our game design is demonstrated in **Figure 6.1** as follows.

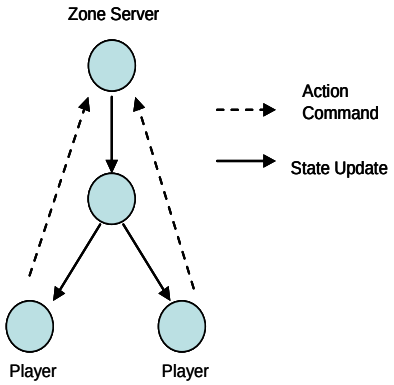


Figure 6.1 Traffic flow

Referring to **Figure 6.1**, the players in a locale are connected to the locale server through the Scribe multicast tree. Game traffic is two directional: action commands from the peers are sent to the locale server directly using unicast; game state updates from the locale server are multicast to peers. The reasons for the two-directional traffic model are two folds. First, there is significant asymmetry between the traffic required by action commands and game state updates. While the action commands only consist of user actions such as movements, the game state updates will include a list of objects for the player's computer to render in the game world. Secondly, client computers usually have an asymmetric capacity with larger download than upload capacity.

In determining the group size of a locale, we consider a trade-off between latency control and bandwidth distribution in Scribe tree. In Scribe, as the tree size increases, the bandwidth stress will be distributed more evenly among nodes. However, it is at the cost of the increased tree length that may negate latency performance. For instance, in our simulation model for the distributed game design, while the multicast tree for Map Type 4 can distribute 4Mbps bandwidth required to support 100 concurrent players, the tree length has averaged to 4 network hops with a maximum tree length of 7. On the other hand, Map Type 1 only distributes a bandwidth of 200kbps, but 90% of the tree

length is equal to or less than 3 and the maximum tree length is 4. In other words, 90% of players in Map Type 1 can reach the locale server in less than 3 network hops, and in the worst scenario, 10% players can reach the server in no more than 4 hops. In Map Type 4, the average and the maximum number of hops to reach a locale server has increased to 4 and 7, respectively. Map Type 1 features player-versus-player fighting, which requires quick actions from players with stringent real-time requirement. Controlling the number of network hops justifies our design for supporting highly real-time sensitive interactivities required for Map Type 1. On the other hand, promoting bandwidth distribution for Map Type 4 justifies our design goal in supporting a large number of concurrent players. Since actions in Map Type 4, the market city, is not as much real-time stringent as in Map Type 1, the increased network hops may be acceptable as far as the game quality is concerned.

6.2.3 Latency Control via Clustered Overlay Setting

Due to the uniform distribution of NodeID, a global Pastry overlay is generally not optimized in terms of topology awareness. As a result, a network hop in a Pastry overlay may travel extra distance, negating latency performance. To improve the latency performance of Pastry overlay, we proposed in Chapter 4 to group nodes that are close to each other in terms of latency into multiple clusters. Instead of having a single global Pastry overlay, based on the groups, we would have multiple clustered Pastry overlays. Since nodes in a clustered overlay are close to each other, and the network hops would not travel across the cluster, latency control will be benefited.

Given the topology on which our simulation is based, we were able to cluster a global overlay of 10000 peers into three local overlays. In the clustered overlay setting, a locale server will first select a proxy node in a clustered overlay that is close to it in terms of latency. Peers in that cluster overlay will then establish a Scribe tree rooted at the proxy node, instead of the locale server as in the global overlay architecture, for game traffic multicast. In short, the game state update traffic follows the path as shown in **Figure 6.2**.

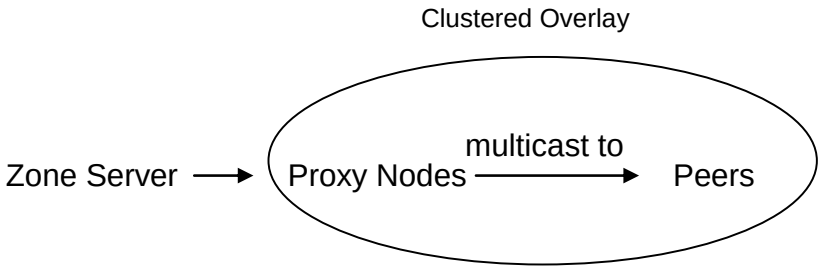


Figure 6.2 Traffic path in the clustered overlay

Our simulation results have shown that, by clustering a global overlay into three clustered overlays, we were able to improve latency performance significantly. For instance, as shown in **Figure 6.3**, while in a global overlay 40% RDP from peers to locale servers are equal to or less than 1.5, the percentage has increased to 80% in the clustered overlay setting, an improvement of 100%. Furthermore, on average, the RDP from peers to locale servers is 1.31 in the clustered overlay setting, versus 1.67 in a global overlay, an improvement of more than 20%. Please refer to Chapter 4 for details.

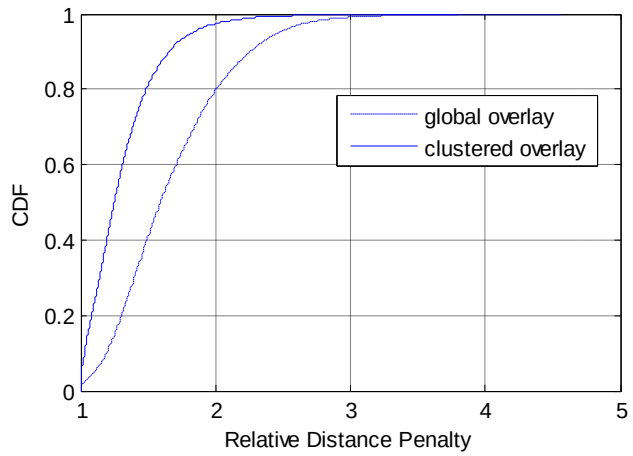


Figure 6.3 CDF of Relative Distance Penalty

To conclude for this section, we have presented a distributed game design that supports MMORPGS using structured P2P overlay. Our main focuses have been on the performance aspects. While the results are promising, we need to further justify the distributed game design with respect to reliability analysis, in response to the churn phenomenon in P2P system. As we shortly discussed in the introduction section, the direct impact of churn is to introduce interruptions in the game sessions, resulting in missed game state updates. In our game design, Map Type 1 would be less resilient to churn since it features quick user action and is highly real-time. For instance, a missed hit by a weapon in a Player-versus-Player environment, which Map Type 1 represents, would lead to completely different game results. On the other hand, Map Type 4 would be much more resilient to churn since it features slow action and is much less real-time. With respect to bandwidth, latency and churn performance requirements, Map Type 2 and 3 are located between Map Type 1 and Map Type 2. Together, those maps will give players a wide selection of features and activities in the game virtual world to explore, and fit a wide area of storylines. It should also be noted that in order to support a capacity of 10,000 concurrent players, a Central Server model would require tremendous resources.

In churn study, we compare the churn impact between different Map Types to justify our distributed game design from the reliability perspective. In addition, we compare the churn impact between the global and the clustered overlay setting, which confirms that the performance gain achieved in the clustered overlay setting is not at the cost of the reliability degradation in the presence of churn.

6.3 Static Analysis – Churn Ripple Factor

We first investigate the multicast tree property in relation to churn. In a multicast tree environment, the failure of an upstream node in a tree will cause disconnections of the downstream nodes from the tree. For multiplayer games, this will lead to potential interrupts of player game sessions. For instance, as shown in **Fig. 6.4,** if the relay node fails, the two players downstream in the tree will be disconnected from the multicast tree for their joined locale.

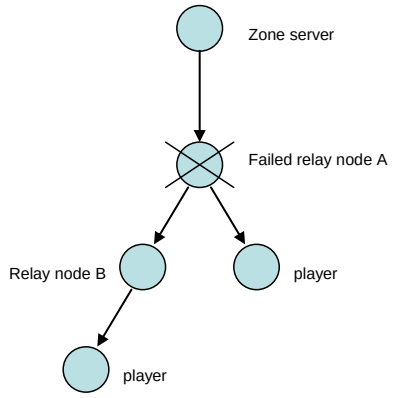


Figure 6.4 A multicast example

In our model, game state updates are propagated from a locale server through a multicast tree to players in the locale. A disconnected player will have to reconnect to the multicast tree to re-establish the communication with the locale server to receive game state updates. While the design of a structured peer-to-peer overlay such as Pastry makes the reconnection process a matter of a small number of overlay hops, the time taken to reconnect to the locale server may lead to missed game state updates to the disconnected players during the reconnection process. How severe the missed game state updates may impact the game quality and user experience is dependent on the game environment. For instance, in a game environment that emphasizes user actions such as a Map Type 1 in our distributed game design, in which a player constantly shoots a weapon to score points, the impact may be significant. However, in a less action demanding environment such as a market city represented by Map Type 4 in our game design, the missed game states may have much less impact on the game quality and user experiences.

We define the number of impacted players by a relay node in a multicast tree as the Churn Ripple Factor (CRF) of that relay node. For instance, in **Fig. 6.4**, the failure of the relay node A will cause three downstream nodes to be disconnected from the multicast tree, and the two of the failed downstream nodes are game players in the multicast tree rooted at the locale server, hence the Churn Ripple Factor (CRF) will be two for relay node A with respect to the locale.

It should be noted that in a peer-to-peer environment, relay nodes are peers who are actual players in the game world. Without incentives from the game provider to entice those nodes to stay in the overlay for longer than their session durations, it should be assumed conservatively that these nodes will leave the overlay when their game sessions are over.

CRF of relay nodes captures the static churn impacts of relay nodes to game players in a specific locale, and hence will assist in establishing an initial evaluation of the system performance in response to churn in our distributed game design.

In analysing CRF, we compare the results between different map types for a particular overlay model, as well as the results based on the same map type but between global overlay and clustered overlay models.

We run the simulation for ten times, and the results are the accumulation of each simulation. For each simulation, we collect relay nodes for each map type and record the Churn Ripple Factor for each relay node. The number of relay nodes in each map type for which we were able to record the CRF is listed in **Table 6.2**.

Table 6.2 Relay nodes recorded for CRF

	Map Type 1	Map Type 2	Map Type 3	Map Type 4
Number of relay nodes in global overlay	29790	30680	23950	21260
Number of relay nodes in clustered overlay	26390	28010	22030	20780

We treat the CRF of each relay node as a random variable, and plot the CDF of that variable. For presentation purposes we only plot Map Type 1 and Map Type 4 for statistical comparison. The plots of Map Type 2 and 3 are located between those of Map Type 1 and Map Type 4. The results are presented in the figures as follows.

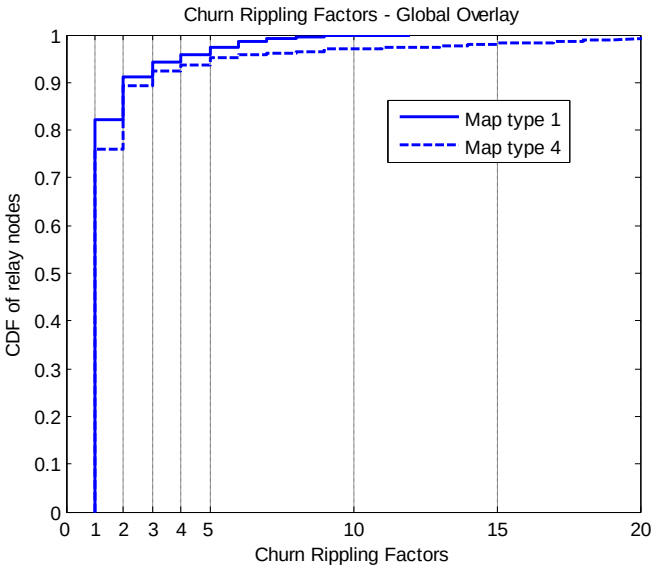


Figure 6.5 Churn Ripple Factors - global overlay

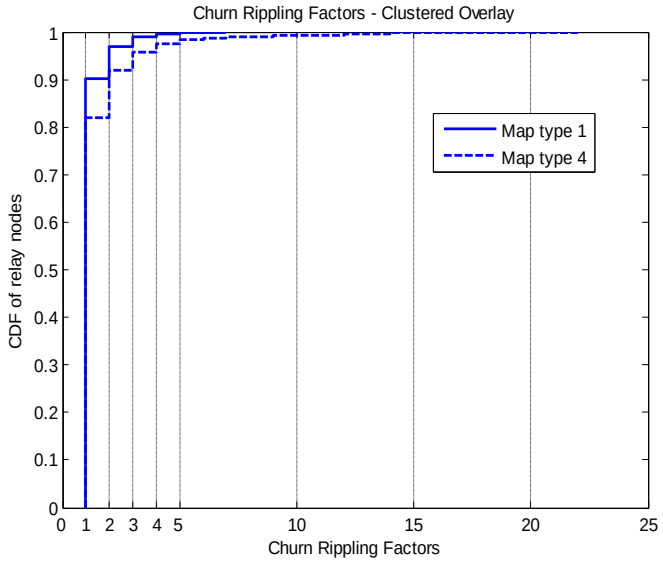


Figure 6.6 Churn Ripple Factors - clustered overlay

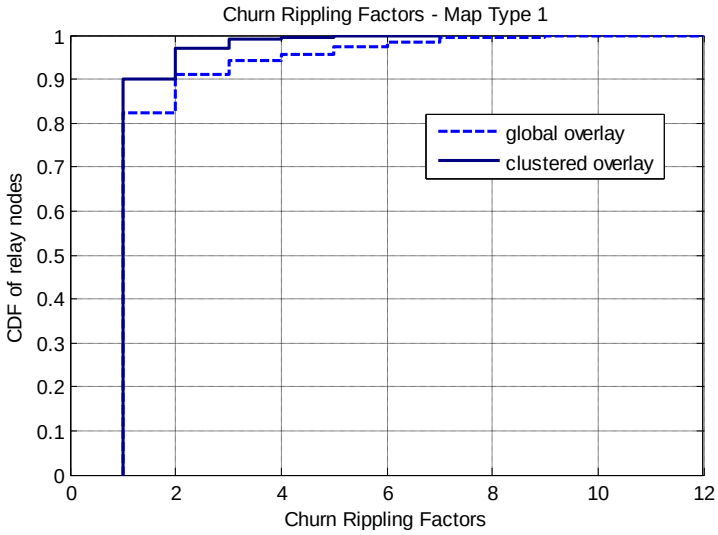


Figure 6.7 Churn Ripple Factors – global vs. clustered overlay – Map Type 1

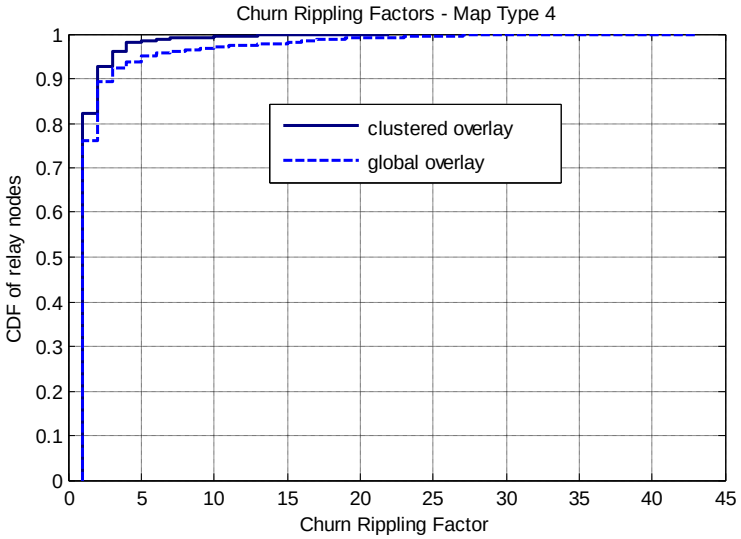


Figure 6.8 Churn Ripple Factors – global vs. clustered overlay – Map Type 4

Table 6.3 CDF of CRF in Global Overlay – Summary for Fig. 6.5

	Map Type 1	Map Type 4
CRF =1	82%	76%
CFR >=2	18%	24%
CRF >=3	9%	11%
CRF >=4	6%	8%
CRF >=5	4%	6%
CRF >=6	3%	5%
CRF >=10	0.17%	3%

In **Figure 6.5** and **Figure 6.6**, we compare CRF between Map Type 1 and Map Type 4 in global and clustered overlay respectively, and both figures show that CRF of Map Type 1 outperforms Map Type 4 with respect to churn. For instance, referring to the second row in Table 6.3 for the statistical summary, in a global overlay 18% of relay nodes in Map Type 1 have CRF >= 2, whereas in Map Type 4, the number has increased to 24%. In other words, statistically, in Map Type 1, the failure of 18% relay nodes will lead to more than 2 players to be dropped from the multicast tree, whereas in Map Type 4, the figure has increased to 24 %, suggesting more players will be prone to interrupts due to the churn of relay nodes. The similar statistical pattern can be found by comparing Map Type 1 and Map Type 4 in the clustered overlay, as plotted in **Figure 6.6**.

It should be noted that Map Type 4 in both **Figure 6.5** and **Figure 6.6** has shown a long tail suggesting a small number of relay nodes have a rather large CRF. In a global overlay as shown in **Figure 6.5** (we did not plot the end of the tail for clarity of presentation), the tail of Map Type 4 has reached the maximum of 43, versus the tail length of 12 for Map Type 1. The long tail of CRF in Map Type 4 can be inspected more clearly in the last row of Table 6.3, which shows that 3% of relay nodes have CRF >=10, whereas in Map Type 1, only 0.17% relay nodes have CRF >= 10. In the clustered overlay model as shown in **Fig. 6.6**, the tail of Map Type 4 has reached 25, versus the tail length of 7 for Map Type 1.

In **Figure 6.7** and **Figure 6.8**, we compare CRF of the same map type between the global and the clustered overlay. Again, for presentation purpose, we only compare Map Type 1 and Map Type 4. Both figures show that CRF of the clustered overlay outperforms the CRF of the global overlay. For instance, as shown in **Figure 6.7**, 10%

of the relay nodes in Map Type 1 have CRF >= 2 in clustered overlay, whereas in the global overlay, the number has increased to nearly 20%.

To summarize the results, if we use CRF as an indication of vulnerability to churn, then Map Type 1 outperforms that of Map Type 4 given a particular overlay model. For the same map type, the churn impact on the clustered overlay is less than that of the global overlay. In other words, with respect to CRF, the clustered overlay outperforms the global overlay.

However, in addition to CRF, session durations of peers would also influence the churn impact. For instance, assume that two relay nodes A and B have the same CRF, but relay node A has a longer session duration than its downstream players' session durations, and relay node B has a shorter session duration than its downstream players' session durations. As a result, relay node A is less likely to cause any interrupts to its downstream players because by the time relay node A leaves the overlay, its downstream players would have completed their game sessions. However, relay node B would cause interrupts to its downstream players because when relay node B leaves the overlay, its downstream players' game session would still be alive.

To further study the churn performance of structured P2P overlay quantitatively, we assign a session time to each node and conduct a discrete event simulation. These will be discussed in the next section.

6.4 Simulation with session times

6.4.1 Simulation Set-up

We choose an exponential distribution to model the session times since it is frequently used in reliability engineering. Furthermore, we assign different mean values based on map type to reflect game features. For instance, players in a battle field (Map Type 1) usually play a quick paced session shooting each other resulting in a short game duration usually less than 30 minutes, whereas in Map Type 4, players may stay much longer in the game for social activities.

The mean of the session times are chosen as 1200, 2400, 3600, 6000 seconds for Map Type 1, 2, 3, and 4 respectively. **Fig. 6.9** shows the distribution of session time durations. Game session duration represents the time duration a player stays in the game from the moment he/she joins the game till the moment he/she leaves the game.

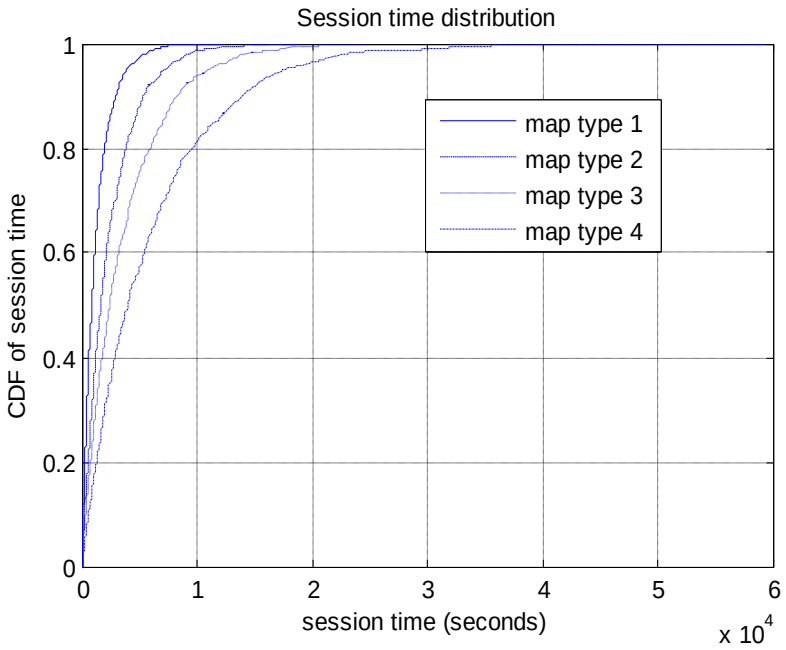


Figure 6.9 Session time distribution

To simplify the modelling and simulation, we assume a small percentage of super nodes in the overlay having indefinite session times, and hence always available. In our simulation model of 10000 nodes, 4% of the overlay population are super nodes, and

assigned to be locale servers or proxy nodes in the clustered overlay configuration. It should be noted that the failure of a locale server node will cause the shutdown of the whole game locale. The game world represented by the locale can not be retrieved unless a backup copy of locale exists in the overlay. For the churn of normal nodes, we would desire to use the underlying P2P overlay resilient design to manage the churn. For instance, if a normal relay node fails, the affected players would only rely on the underlying P2P protocol to reconnect to the locale server, and the cost is small. However, if a super peer fails, the impact would be too severe to be handled by the underlying P2P resilience, and would require a backup management protocol. Ideally, the backup management protocol should allow the transfer of the locale server to be transparent to the players in the locale. The design and evaluation of such a scheme remains as our future work. It should be noted that our assignment of only a small percentage of super nodes would benefit the feasibility and design of the management protocol.

After assigning the initial session time to each node, we set the simulation resolution at one second and start the simulation clock at time zero. For each second progressed by the simulation clock, we capture the number of nodes disconnected from the overlay due to their session time out at that time instant. If we define the failure rate as the number of nodes disconnected from the overlay due to session duration time out at each second, the CDF plot of failure rate is shown in **Fig. 6.10**. On average, the failure rate is four nodes per second in our model.

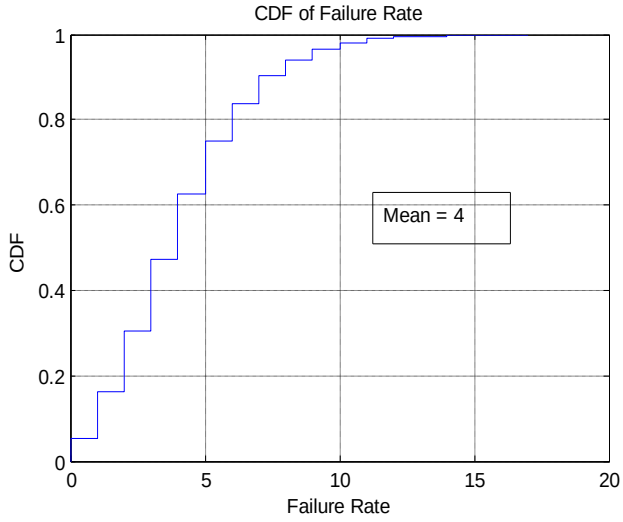


Figure 6.10 Node failure rate

For each failed node, if the node is a relay node in a multicast tree, we capture the number of player nodes in the map that depend on this relay for their state update information from the locale server. In other words, we capture the CRF of that relay node. Those player nodes will obviously be dropped off from the multicast tree at this time instant due to the failure of the relay node, and each of them will register an interrupt in their current session. We record the interrupt occurrences for each player in his/her session duration. The dropped player nodes then rejoin the tree rooted at the locale server. To keep the total number of players in the game more or less constant, for every failed node, a fresh node will join the overlay following the Pastry joining protocol. We run the simulation clock for 3600 seconds and the results are presented in the next section.

6.4.2 Simulation Results

For simulation results, we are particularly interested in the number of interrupts that each player has encountered during his/her session. In a peer-to-peer environment, those interrupts are the direct results of the churn of peers, which is inevitable due to the nature of P2P system. As a result, when investigating the churn impact, we mainly analyse the statistical results of interrupt count in player game sessions. In other words, we use Interrupt Count as a metric to measure the churn impact or resilience of the P2P system.

We should also keep in mind that a given game may have different level of tolerance for interrupts. In our game design as we discussed in the previous sections, we would desire less interrupts for Map Type 1, whereas Map Type 4 may not be as sensitive to interrupts.

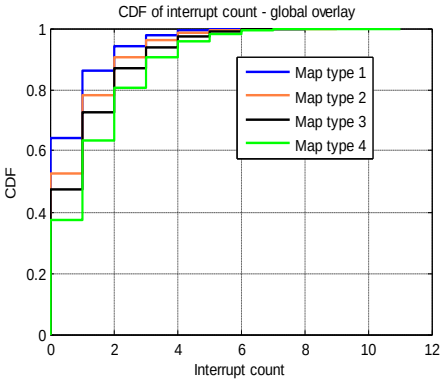
We run the simulation for ten times, and the number of game sessions we are able to collect based on Map Type is listed in **Table 6.4** as follows. We compare the results between different map types in a particular overlay model, as well as, the results based on the same map type but between different overlay models.

Table 6.4 Number of sessions collected

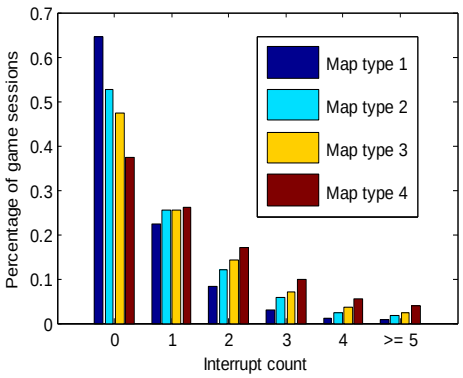
	Map type 1	Map type 2	Map type 3	Map type 4
Sessions collected	91400	66720	46920	37410

6.4.2.1 Comparison between Map Types

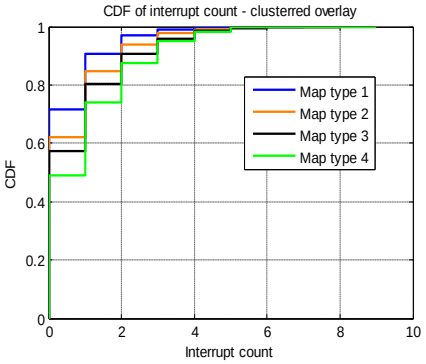
We take the number of interrupts or interrupt count in each player game session duration as a discreet random variable. We first compare the CDF of interrupt count between different map types in global and clustered overlay, respectively. The results are presented in the **Figure 6.11** (a)-(d).



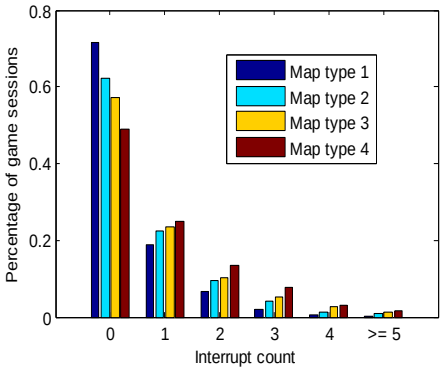
(a) CDF of Interrupt Count – the global overlay



(b) Histogram of Interrupt Count – the global overlay



(c) CDF of Interrupt Count – the clustered overlay



(d) Histogram of Interrupt Count – the clustered overlay

Figure 6.11 Comparison of Interrupt Count between Map Types

Table 6.5 Summary of Fig. 6.11(a) – the global overlay

	Map Type 4	Map Type 3	Map Type 2	Map Type 1
Session percentage with interrupt count = 0	37%	47%	53%	65%
Session percentage with interrupt count >= 1	63%	52%	47%	35%
Session percentage with interrupt count >= 2	36%	27%	22%	13%
Session percentage with interrupt count >= 3	19%	14%	10%	5%
Session percentage with interrupt count >= 4	10%	6%	4%	2%
Session percentage with interrupt count >= 5	4%	2.5%	1.6%	0.8%

Table 6.6 Summary of Fig. 6.11(c) – the clustered overlay

	Map Type 4	Map Type 3	Map Type 2	Map Type 1
Session percentage with interrupt count = 0	49%	57%	62%	72%
Session percentage with interrupt count >= 1	51%	43%	38%	28%
Session percentage with interrupt count >= 2	26%	19%	15%	9%
Session percentage with interrupt count >= 3	13%	9%	6%	3%
Session percentage with interrupt count >= 4	5%	4%	2%	1%
Session percentage with interrupt count >= 5	1.7%	1.4%	0.9%	0.3%

Both **Figure 6.11**(a) and **Figure 6.11**(c) together with their associated histograms show a significant improvement of resilience to churn as the Map Type index decreases. For instance, as summarized in **Table 6.5** for **Fig. 11**(a), for the global overlay, 37 percent of the player game sessions in Map Type 4 have an Interrupt Count of zero. In other words, 37 percent of players playing in Map Type 4 are able to complete their game sessions without the impact of churn in the P2P system. As the map index decreases, the percentage of game sessions with zero Interrupt Count has improved consistently to 47 percent for Map Type 3, 53 percent for Map Type 2 and 65 percent for Map Type 1. The step of improvement is approximately 10 percent for each decrement of map index.

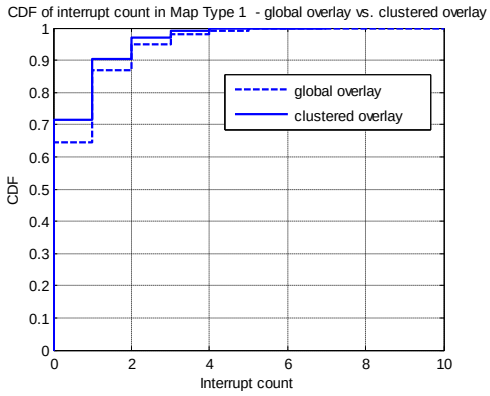
By further inspecting the histogram in **Fig. 11**(b), it can be seen that for each discrete value of Interrupt Count greater than one, the percentage of game sessions subject to that value increases as the Map Type Index increases. In other words, as Map Type index increases, more player game sessions will be subject to more interrupt occurrences. This trend can be seen more clearly in **Table 6.5**. For instance, as listed in the second row of **Table 6.5**, 63% of the game sessions are subject to Interrupt Count >= 1 in Map Type 4, whereas in Map Type 3,2, and 1, the percentage has decreased consistently to 52, 42 and 35, respectively, with approximately 10 percent

decrease or churn resilience improvement for each decrement of Map Type index. By comparing directly between Map Type 1 and Map Type 4, the significance of the churn resilience improvement as the Map Type Index decreases can be seen more clearly. It should also be noted that, the amount of improvement increases when the discrete value of interrupt count increases. For instance, referring to **Table 6.5**, the number of sessions subject to interrupt count >= 1 has decreased from 63% to 35% from Map Type 4 to Map Type 1, with 27% improvement; whereas for interrupt count >=3, the result has decreased from 19% for Map Type 4 to 5% for Map Type 1, with nearly an improvement of two folds. It should be noted that, while 4% of the game sessions have interrupt count >=5 in Map Type 4, the number has dropped below 1% for Map Type 1.

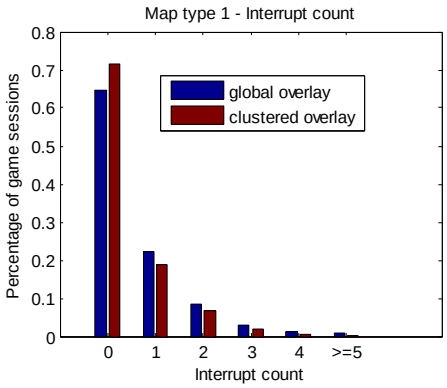
We also compared the churn resilience between different map types in clustered overlay architecture and the results are presented in **Fig. 11**(c) and **Fig. 11**(d) with summarized statistical information in **Table 6.6**. While differing in values from the results found in a global overlay model, the general trend in churn performance improvement as the Map Type Index decrements is consistent to those found in the global overlay setting. For instance, as summarized in **Table 6.6**, while 49% the game sessions in Map Type 4 have an Interrupt Count of zero, the percentage has increased/improved to 57, 62, and 72 for Map Type 3, 2, and 1 respectively. The improvement is approximately 10% for each decrement of Map Type Index. Furthermore, the amount of improvement increases when the discrete value of Interrupt Count increases. For instance, referring to **Table 6.6**, the number of sessions subject to Interrupt Count >= 1 has decreased from 51% to 28% from Map Type 4 to Map Type 1, with 21% improvement. For interrupt count >=3, the result has decreased from 13% to 3%, an improvement of more than 2 folds. Finally it should be noted that, while 1.7% of the game sessions have Interrupt Count >= 5 in Map Type 4, the number has dropped to 0.3% for Map Type 1.

6.4.2.2 Comparison between the overlay architectures

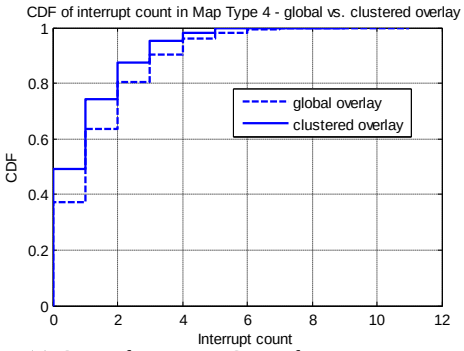
We further compare the churn resilience between the global overlay and the clustered overlay architectures for a particular Map Type. For presentation purpose, we only compare Map Type 1 and Map Type 4 for each. The results are plotted in **Figure 6.12**, and the statistical summary is listed in **Table 6.7** and **Table 6.8** as follows.



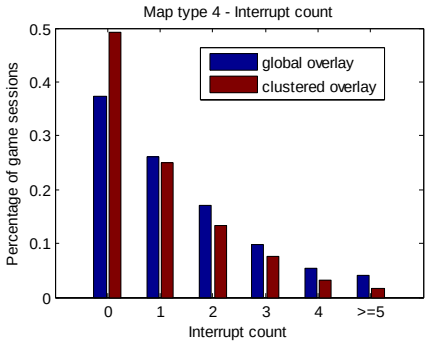
(a) CDF of Interrupt Count for Map Type 1



(b) Bar plot of Interrupt Count for Map Type 1



(c) CDF of Interrupt Count for Map Type 4



(d) Bar plot of Interrupt Count for Map Type 4

Figure 6.12 Comparison of Interrupt Count between the overlays

Table 6.7 Summary of Fig. 6.12 (a) and Fig. 6.12 (b) – Map Type 1

	global overlay	clustered overlay	improvement
Session percentage with interrupt count = 0	65%	72%	7%
Session percentage with interrupt count >= 1	35%	28%	7%
Session percentage with interrupt count >= 2	13%	9%	4%
Session percentage with interrupt count >= 3	5%	3%	2%
Session percentage with interrupt count >= 4	2%	1%	1%
Session percentage with interrupt count >= 5	0.8%	0.3%	0.5%

Table 6.8 Summary of Fig. 6.12 (c) and Fig. 6.12 (d) – Map Type

	global overlay	clustered overlay	improvement
Session percentage with interrupt count = 0	37%	49%	12%
Session percentage with interrupt count >= 1	63%	51%	8%
Session percentage with interrupt count >= 2	36%	26%	10%
Session percentage with interrupt count >= 3	19%	13%	6%
Session percentage with interrupt count >= 4	10%	5%	5%
Session percentage with interrupt count >= 5	4%	1.7%	2.3%

Figure 6.12 (a) - (d) show that the clustered overlay outperforms the global overlay with respect to churn resilience. For instance, for Map Type 1, as summarized in **Table 6.7** for **Fig. 6.12** (a) and (b) , while 65% of the game sessions in the global overlay have Interrupt Count = 0, this percentage has improved to 72% in the clustered overlay. Furthermore, the churn resilience improvement is consistent for each discrete value of the Interrupt Count. For instance, as listed in **Table 6.7**, while 35% of the game sessions in the global overly are subject to Interrupt Count >= 1, the number has dropped to 28% in the clustered overlay.

For Map Type 4, the churn resilience improvement in the clustered overlay over the global overlay is similar to that of Map Type 1. It should be noted that, for each discrete value of Interrupt Count, the improvement amount increased more noticeably than that of Map Type 1. For instance, as listed in **Table 6.8**, while 37% game sessions have an Interrupt Count = 0 in the global overlay, the percentage has improved to 49% in the clustered overlay. The improvement is 12%, compared to 7% for Map Type 1 as listed in **Table 6.7**.

6.4.2.3 Overlay Maintenance Cost Due to Churn

The cost of churn to the overlay is mainly the running of the maintenance protocol, since nodes in the overlay contain other nodes as entries in their routing table, leaf set and neighbour set. We finally plot the CDF of percentage of nodes that require maintenance during the entire simulation duration, and the results are shown in **Fig. 6.13**.

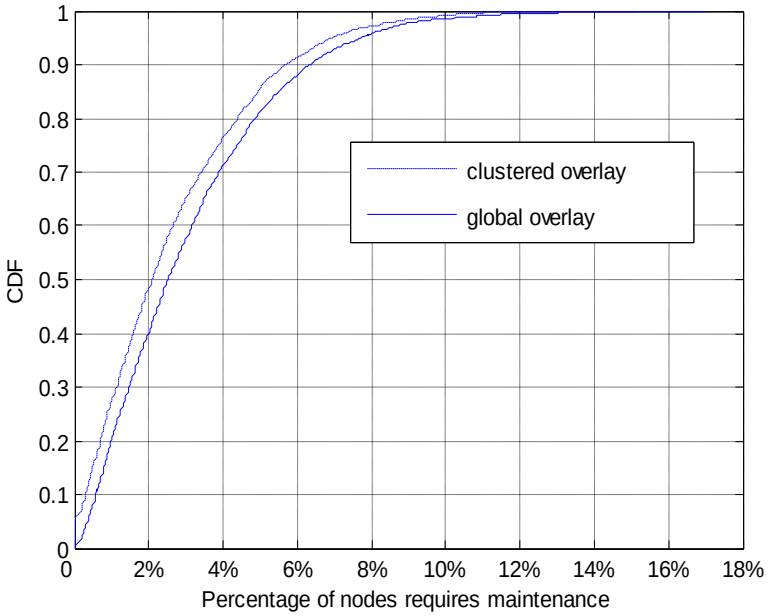


Figure 6.13 Maintenance cost

Given our game design, it is shown in **Figure 6.13** that the overlay maintenance cost due to churn is low. For instance, in the global overlay, more than 95% of the time during the simulation duration, less than 8% of the nodes in the overlay require maintenance to refresh some entries in the routing table due to the churn. The maintenance cost due to churn in the clustered overlay is marginally smaller than that of the global overlay, as shown in **Figure 6.13**.

6.4.3 Discussions

In addition to the trade-off between latency and bandwidth distribution as we discussed in Section 6.2.2, the simulation results have demonstrated that a further trade-off is required between the game features and the churn resilience. In our distributed game design, as the Map Type Index increases, the player group size increases resulting in a multicast tree with more members. The underlying structured P2P overlay is able to distribute the bandwidth stress more evenly among the nodes in the tree established to be able to support a large number of concurrent players. However, the cost of this improved bandwidth distribution capacity is an increase of latency and degradation of churn resilience. Reflected in our simulation results, Map Type 1 has a much better churn resilience than that of Map Type 4.

However, it should be noted that the degradation of churn resilience with the increase of Map Type index is much slower than the increase of player group size. From Map Type 1 to Map Type 4, the player group size has increased five times from 20 to 100 players per group. However, the churn resilience has only decreased by less than 30% in both the global overlay and the clustered overlay architectures. This is apparently a positive factor as the large groups sizes, which aim at social interaction, may also be less sensitive to interrupts due to churn.

While the interrupts caused by churn are inevitable in a P2P environment, it remains a question as to how many interrupts is acceptable before the game quality deteriorates. While the answer to this question is generally related to the nature and the level of real-time interactivity required by the game, it should be noted that many techniques are frequently used to compensate for delayed or missed game states [12]. As a rule of thumb, let us assume conservatively that two interrupts per session is the threshold for players to consider the quality unacceptable in Map Type 1. Then based on **Fig. 6.12 (a)** , 95% and 97% of game sessions will be under the threshold for the global and the clustered overlays respectively. For Map Type 4, if we assume a threshold of four interrupts per session as the acceptable level of quality, then based on Fig. 6.12(c), 95% and 98.3% of the game sessions will be under the threshold for the global and the

clustered overlay respectively. Based on these figures, it is our opinion that structured P2P is a promising infrastructure in supporting MMORPGs, as far as churn is concerned.

It should be noted that, in comparison with the global overlay, the clustered overlay has achieved performance gain with respect to both latency and churn. Furthermore, the performance gain is achieved without extra overlay maintenance cost. The performance gain of the clustered overlay architecture is summarized in **Table 6.9** as follows.

Table 6.9 Summary of performance gain for clustered overlay

	Global overlay	Clustered overlay	Performance gain
Average RDP	1.67	1.31	0.34
Session percentage with interrupt count = 0 (Map Type 1)	65%	72%	7%
Session percentage with interrupt count = 0 (Map Type 4)	37%	49%	12%
8^{th} percentile maintenance cost	96%	97.50%	1.50%

6.5 Related Work

Works related to churn in structured P2P overlay [73-77] generally address the query performance in the overlay under churn, and propose techniques related to overlay maintenance protocol to improve the query performance. For instances, Li et al. [75] proposed to tune the parameters of DHT protocol to achieve a high query success ratio under constant churn. Hoang et al. [74] proposed to add an extra layer of maintenance protocol enforced during a node joining or leaving the overlay, and demonstrated that the query success ratio improved consistently under different churn rate comparing to that of a DHT overlay without the enforced maintenance protocol.

Of relevance to our work, it should be noted that a dropped node from the multicast tree is reconnected to the tree through the overlay query protocol, whose performance is the fundamental design goal of structured P2P overlay. Gunmmadi et al. [78] show that for a ring based DHT overlay such as Pastry, even after 30% of the links are broken, there are still connected paths between all node pairs, suggesting the overlay sustains a high query success ratio even under severe churn. Furthermore, Li et al. [75] show that if sufficient bandwidth is allocated for running maintenance protocol (20bytes/node/s in

the simulation), a less than 200ms query latency can be achieved. Based on these works, we expect a high success ratio and low cost with respect to latency for the reconnection process of a dropped node in our distributed game design. Furthermore, while the abovementioned works can potentially improve the reconnection process for a game session, it should be noted that our distributed game design does not exclude the involvement of a central service for overlay management related work. For instance, in the worst scenario when a dropped node fails to reconnect to the multicast tree to receive game updates due to the failure of the DHT maintenance protocol, a central service can guide the node to connect to the locale server directly. We believe the contingency service is important given the stringent Quality of Service expected for game playability.

To our best knowledge, we are the first in investigating the churn resilience of a structured P2P overlay for supporting a distributed game design for MMORPG as of the writing of this Chapter.

6.6 Conclusions

In this chapter, we focus on the impact of churn– the continuous process of node arrival and departure – on the P2P system to support MMORPGs. We recognize that the churn can potentially result in interrupts to otherwise contiguous game sessions. We use the Interrupt Count in the game sessions as the evaluation metric to study the churn performance of the game.

Our simulation has shown that, in our distributed game design, as the real-time stringency of the map increases, and group size decreases, the churn resilience shows a consistent improvement. For instance, in a standard global Pastry overlay, while 37% of nodes in Map Type 4 are subject to Interrupt Count = 0 in their game sessions, the number has improved to 65% for Map Type 1. In other words, 37% and 65% of nodes in Map Type 1 and Map Type 4, respectively, can complete their game sessions without experiencing the impacts of churn. Furthermore, for each discrete value of Interrupt Count, Map Type 1 has shown a consistent improvement over Map Type 4. The churn resilience of Map Type 2 and 3 are located between Map Type 1 and Map Type 4.

We further experimented with a clustered overlay architecture that clusters nearby nodes to address the performance issues of a global overlay. We show that the clustered overlay achieves an overall performance gain with respect to both latency and churn. Simulation results also show that the performance achieved by the clustered overlay does not lead to extra cost with respect to overlay maintenance.

With respect to churn resilience, the structured P2P overlay appears a promising and reliable infrastructure to support MMORPGs. If we assume that an Interrupt Count of <= 2 and 4 for an average session are the thresholds for the acceptable quality-of-service for Map Type 1 and 4 respectively, then our simulation results have shown that more than 95% of game sessions can achieve the quality-of-service in both the global overlay and the clustered overlay architectures.

It should also be noted that 'churn behaviour' can be impacted by many factors such as game feature, user behaviours and network conditions. For instance, a network outage may lead to all the users in that network go offline contributing to churn from a game's perspective. In this chapter, to conduct the dynamic analysis of churn, we focus on a normal user behaviour – user goes offline when his/her session time in the game is completed. We believe the study establishes a useful baseline to allow further study of churn when considering other factors such as network breakdown.

Finally, it should be noted that our churn study has utilised a generic model that is widely used in the reliability enginnering – an exponentially distribution for the user session time to model the churn. While a realistic churn pattern is difficult to capture in a real MMORPG, it may produce a valuable futurework to futher justify the performance of peer-to-peer network in supporting MMORPGs.

Chapter 7. Conclusions

7.1 Overview

The resource constraints of Central Server model have hindered the scalability of online games with respect to the number and geographic span of concurrent players that can be supported. As an alternative to Central Server Model, a P2P system addresses the scalability bottlenecks of Central Server Model by leveraging the resources of participating peers. Since the resources – including CPU cycles, data storage (e.g., RAM and hard disk), and bandwidth – are contributed by participating users, a P2P system could ideally achieve organic growth by expanding arbitrarily without the requirement of a "forklift upgrade" to the existing infrastructure, for instance, the replacement of a central server with more powerful hardware.

In this thesis, we have investigated the issues related to a peer-to-peer infrastructure in supporting Massively Multiplayer Online Role Playing Games (MMORPGs). We focus on two significant aspects, *performance and reliability*, to establish the feasibility of P2P system for this purpose.

With respect to performance, we concentrate on bandwidth and latency management. This is to address that, in a P2P system, peers usually have limited bandwidth capacity, and are far apart from each other in terms of latency. With respect to reliability, we focus on the study of churn – the continuous process of peer arrival and departure in the P2P overlay, and its impact on the performance of MMORPGs. We now provide a summary of the main contributions and recommendations of this thesis, as well as, future works that are conceived on the basis of knowledge and insights gained from this research.

7.2 Summary of Contributions and Recommendations

7.2.1 Bandwidth Management

To address the bandwidth limitations of peers in a P2P system, we proposed in Chapter 3 a model that uses P2P multicast for dissemination of state updates from the locale servers to clients. The multicast scheme is based on Scribe, a multicast tree that is built on top of Pastry, a typical structured P2P system. We take advantage of the design of Pastry for its inherent P2P characteristics such as decentralized control, self-organization, adaptation and scalability. To evaluate the performance of the traffic model in supporting online games, we developed a distributed game design that represents an advanced MMORPG configuration with respect to game traffic pattern. We then integrated the traffic model into the game design to evaluate the performance of the P2P system in supporting the game. The results of a simulation based on a Pastry overlay of 5000 nodes have shown that the Scribe multicast traffic model performes well in supporting the game by effectively distributing the bandwidth required for the game traffic among peers in the overlay. Furthermore, through the bandwidth distribution among the peers, the traffic model has effectively limited the upstream bandwidth stress on peers to feasible values, for example less than 250kbps. In contrast, if a unicast model were used to support the same game design, the bandwidth incurred at the server side would be far beyond the capacity of a typical peer (e.g. a household computer with high end ADSL link).

In Chapter 3, we also identified a potential bottleneck in that a small number of nodes in the multicast tree, in particular, those that serve as the roots of multicast trees may be subject to relatively heavy bandwidth stress. We hence developed two low-cost techniques to remove the bottleneck. It should be noted that these techniques take advantage of the adaptive design of the underlying P2P overlay.

Moreover, in Chapter 3, through the performance evaluation of Scribe in supporting a MMORPG game, we demonstrated that a trade-off is required between bandwidth and latency for the map design in MMORPG. For instance, in our simulation model for the distributed game design, while the multicast tree for Map Type 4 can effectively

distribute bandwidth required to support 100 concurrent players, the tree length has averaged to four network hops with a maximum tree length of 7. On the other hand, Map Type 1 only distributes a bandwidth required to support 20 concurrent players, but 90% of the tree length is equal to or less than three and the maximum tree length is four. As a result, we recommended that, while Map Type 1 has a small group size of 20 concurrent players, it is better for real-time stringent interactivities due to better latency performance. On the other hand, Map Type 4 is able to support a larger number of concurrent players by being able to distribute more bandwidth stress among peers. However, the cost is the reduced latency performance. Hence, Map Type 4 is better suited for game scenes such as a market city where the game will become more interesting with a large number of players, but the interactivities between players can be relaxed with respect to real-time stringency.

In Section 4.4.4 in Chapter 4, we further proposed a two-level overlay architecture to improve the bandwidth management of the P2P overlay in supporting MMORPGs. In this model, the higher level overlay is established among the nodes located close to network access points, and the lower level among the nodes in a local domain. In this case, the bandwidth stress required by applications can now be distributed among the nodes in the higher level overlay, as well as, in the lower level. We demonstrated that with the improved bandwidth performance of the two-level overlay, the number of concurrent players in a game map whose traffic pattern is projected from the game design can be pushed up to more than two hundred, while maintaining the bandwidth stress on peers at a feasible level, e.g., less than 1Mbps. It should be noted that if the same number of concurrent players in the game map is supported using a unicast model, the bandwidth stress incurred on peers would be prohibitively high, e.g., up to 24Mbps, beyond the capacity of a common household computer.

7.2.2 Latency Management

In Chapter 4, we argue that a Scribe multicast tree built on top of Pastry is not optimal with respect to latency performance. We identify that the final hop of a Scribe traffic path is topologically randomised and makes the largest contribution to the latency of delivering the message. We hence developed a heuristic to improve the latency performance of Scribe. The heuristic involves two techniques. The first technique groups nodes that are close to each other in terms of latency into multiple clusters. Instead of having a single global Pastry overlay, we would have multiple clustered Pastry overlays. Since nodes in a clustered overlay are close to each other in terms of latency, and the network hops would not travel across the cluster, latency control would be improved. The second technique applies Proxy Neighbour Selection, a light weight technique to the final hop in the traffic path. In the clustered overlay model, a locale server hosting a map in a MMORPG will first select a proxy node in a clustered overlay that is close to it in terms of latency. Peers in that clustered overlay will then establish a Scribe multicast tree rooted at the proxy node. The hop from the proxy node to the locale server will become the last hop in the traffic path from a peer in the cluster to the locale server, and the application of Proxy Neighbour Selection to the last hop will ensure the last hop latency is optimal from the peer's perspective. We demonstrated through simulation that the techniques have improved the latency performance of Scribe significantly. On average, the RDP (a measure of routing performance, RDP = 1 is optimal) from peers to locale servers is 1.31 in a three clustered overlay, versus 1.67 in a global overlay, an improvement of more than 20%. In a twenty-four clustered overlay model (a global overlay is clustered into twenty four clustered overlays), the RDP has further improved to 1.1, and is nearly optimal.

In Chapter 5, we introduced the concept of a seamless map for MMORPGs. The general approach for MMORPG to achieve scalability is to partition the game world into smaller components called locales. In a Central Server model, the simulation of those locales can be distributed among the servers in a server cluster, whereas in a P2P system, those locales can be assigned to the peers who are willing to take up the roles of locale servers.

While those locales in a game map are segregated from each other with no communication required between the locale servers, games usually rely on certain techniques to hide the segregation between locales from the players. For instance, a player has to fly through a valley or walk through a path to be disconnected from one locale and join another. In a seamless map, however, the boundaries between locales become truly transparent to players without the need of the abovementioned techniques. This is critically important for a P2P architecture, as the locale servers have limited capacity and bandwidth and, hence, locales are likely to be rather small. We envisage a seamless map will bring more exciting features to games. For instance, a battlefield that engages a large number of players could adopt a seamless map that gives the players the impression of a contiguous virtual space without boundary limitations.

A seamless map requires the communication between locale servers to allow the interactions between players across the neighbouring locales. In Chapter 5, we argue that the inter-locale communication cost is not a critical issue in Central Server model since servers are usually co-located and interconnected over a high-speed LAN environment. However, in a P2P system, it is critical to seek approaches to reduce the inter-locale communication cost since locale servers are assigned to peers that can be far from each other in terms of latency. To that end, we formulate a Locale Assignment Problem that aims to minimize the inter-locale communication cost in a P2P system, and demonstrate through an optimisation model that the problem is NP-hard. We then propose a low-cost heuristic for the Locale Assignment Problem. The key aim of the heuristic is to partition the physical network into bins so that peers in a common bin are close to each other in terms of latency proximity, as well as, aggregate the partitions of a game virtual map by clustering neighbouring locales in the map. We then demonstrate that by assigning clustered neighbouring locales in the virtual map onto peers in a common physical network bin, we are able to achieve similar inter-locale communication cost for the Locale Assignment Problem as that can be achieved by the optimization model we formulated, while avoiding the prohibitively high computation cost required by the latter.

It should also be noted that the key to the algorithm and technique developed in Chapter 5 is to relate locale, a game virtual world, to node clusters that is facilitated by clustering techniques.

7.2.3 Churn Resilience Study

While Chapter 3, 4 and 5 focus on the *performance* aspect of a P2P system in supporting MMORPGs, Chapter 6 concentrates on the *reliability* aspect to consolidate the feasibility of the P2P system in supporting MMORPGs. In a P2P system, reliability generally refers to how system performs in response to churn – the continuous process of peer arrival and departure.

Given the multicast traffic model and distributed game design we have studied in Chapter 3 to 5, we recognize that the direct impacts of churn in the game design are potential interrupts introduced to otherwise contiguous game sessions. For instance, if a relay node in a multicast tree leaves his/her game session, the downstream nodes of the relay nodes in the multicast tree will be disrupted in receiving game state updates from the locale server, the root of the corresponding multicast tree. As a result, we use the interrupt count in game sessions as the evaluation metric to study the churn resilience of P2P system in supporting the game design.

To quantitatively study the churn resilience of the P2P system in supporting our game design, we assign a probabilistic game session time to each peer in the overlay in order to model the game session duration of the peer. We choose exponential distribution to model the session times since it is frequently used in reliability engineering. Furthermore, we assign different mean values of game session duration based on map types to reflect game features. For instance, players in a battle field (Map Type 1) usually play a quick paced session shooting each other resulting in a short game duration usually less than 30 minutes, whereas in Map Type 4, players may stay much longer in the game for social activities.

The results achieved through a discrete simulation have shown that a trade-off is required between game features with respect to real-time stringency, and the churn resilience. In our game design, as the group size decreases, the churn resilience shows a

consistent improvement. For instance, in a standard global Pastry overlay, while 37% of nodes in Map Type 4 are subject to interrupt count = 0 in their game sessions, the number has improved to 65% for Map Type 1. In other words, 37% and 65% of peers in Map Type 1 and Map Type 4, respectively, can complete their game sessions without the impacts of churn. Comparing to Map Type 4, Map Type 1 has improved nearly 30% with respect to churn resilience. Recall that churn may lead to missed game state updates that potentially negate game quality. In our game design, Map Type 1 would be less resilient to churn since it features quick user action and is highly real-time. For instance, a missed hit by a weapon in a Player-versus-Player environment, which Map Type 1 represents, would lead to completely different game results. Hence, it is desirable to have better churn resilience for Map Type 1. On the other hand, Map Type 4 would be much more resilient to churn since it features slow action and is much less real-time stringent. Hence, while allowing a large group size to support game features such as market city, Map Type 4 may tolerate poorer churn resilience in comparison to Map Type 1. Game providers should consider this trade-off in designing virtual world maps in a P2P environment.

We also compared the churn resilience between the clustered overlay and the global overlay. The results achieved through the discrete event simulation have shown that the clustered overlay outperforms the global overlay with respect to churn resilience. For instance, for Map Type 1, while 65% of the game sessions in the global overlay setting have interrupt count = 0, the percentage has improved to 72% in the clustered overlay setting, an improvement of 7%. Furthermore, the simulation results also show that the performance gain achieved by the clustered overlay setting does not incur extra cost to overlay maintenance.

Finally, the simulation results have shown that the overall churn resilience of structured P2P overlay in supporting MMORPGs is promising. If an interrupt count of a game session <= 2 is the threshold for the quality-of-service for Map Type 1, and an interrupt count of a game session <= 4 is the threshold for the quality-of-service for Map Type 4, our simulation results have shown that more than 95% game sessions can achieve the quality-of-service in both the global overlay setting and the clustered overlay setting.

7.3 Future Works

The work presented in this thesis could be extended or improved along the following research directions which are conceived on the basis of the knowledge and insights gained from this research:

In both Chapter 4 and 5, we have used a GT-ITM Transit-Stub topology to model the overlay network in supporting our distributed game design. While the topology provides a reasonable platform for modelling, it would be interesting to see how the P2P overlay performs in supporting our game design in a more realistic network environment such as PlanetLab[5]. PlanetLab provides an infrastructure consisting of nearly a thousand nodes across the globe for academic research. An experiment of our game design on PlanetLab infrastructure would provide an opportunity to further investigate the feasibility and performance of P2P system in supporting MMORPGs.

In Chapter 4 and 5, to improve the latency performance of the P2P overlay in supporting the game design, we have relied on the structure of GT-ITM Transit-Stub topology for grouping nearby nodes into clusters for modelling purpose. While techniques [43, 55] are available to effectively cluster nodes in terms of latency, we envisage that game distributed design might have specific needs for clustering techniques. For instance, while the landmark binning scheme [55] can effectively group nodes that are close to each other, the number of bins can not be easily controlled. As a result, when the number of landmarks increases, clustering heuristic may generate too many bins with each bin having only a few nodes in it. We believe that the study of game specific needs for clustering techniques may become an interesting area to explore.

In Chapter 6, to simplify the modelling of a churn study, we assume that a small percentage (4%) of super nodes are always available and do not undergo churn. Those super peers are the roots of multicast tress that are locale servers in MMORPGs. It should be noted that the failure of a locale server node will cause the shutdown of the whole game locale. The game world represented by the locale can not be retrieved unless a backup copy of locale exists in the overlay. For the churn of normal nodes, we

[5] http://www.planet-lab.org

would desire to use the underlying P2P overlay resilient design to manage the churn. For instance, if a normal relay node fails, the affected players would only rely on the underlying P2P protocol to reconnect to the locale server, and the cost is small. However, if a super peer fails, the impacts would be too severe to be handled by the underlying P2P resilient design, and would require a backup management protocol. Ideally, the backup management protocol should allow the transfer of the locale server to be transparent to the players in the locale. The design and evaluation of such a back up protocol would be interesting for the future works.

Finally, it should be emphasised that the purpose of this thesis is to conduct a feasibility study of using P2P network in supporting online games, focusing on two aspects – performance and reliability. It should be recognised that there are still some challenges to apply the techniques and theories developed in the thesis to a practical online game. For instance, how to facilitate the efficient communication for peers behind NAT devices. Practical aspects such as this will remain an interesting research area for P2P network in supporting online games.

Bibliography

1. *2010 Essential Facts About the Computer and Video Game Industr*: http://www.theesa.com/facts/pdfs/ESA_Essential_Facts_2010.PDF.

2. Chambers, C., W.-c. Feng, M. Sambit Sahu, M. Debanjan Saha, and D. Brandt, *Characterizing Online Games.* IEEE/ACM Transactions on Networking (TON), 2010. 18(2).

3. *EVE Online*: www.eveonline.com.

4. Blizzard, www.blizzard.com.

5. Counter-Strike: www.counter-strike.com.

6. NCsoft: www.lineage.com.

7. Greenhalgh, C. and S. Benford, *MASSIVE: a collaborative virtual environment for teleconferencing.* ACM Transaction on Computer-Human Interactions (TOCHI), 1995. 2(3): p. 239-261.

8. Safaei, F., P. Boustead, C.D. Nguyen, J. Brun, and M. Dowlatshahi, *Latency-Driven Distribution: Infrastructure Needs of Participatory Entertainment Applications*, in *IEEE Communication Magazine*. 2005. p. 106-112.

9. Pantel, L. and L.C. Wolf. *On the Impact of Delay on Real-Time Multiplayer Games*. in *Proc. of the 12th int. workshop on Network and Operating systems Support for digital audio and video*. 2002. Miami, Florida, USA.

10. Vaghi, I., C. Greenhalgh, and S. Benford. *Coping with Inconsistency due to Network Delays in Collaborative Virtual Environments*. in *Proc. of the ACM symposium on Virtual Reality Software and Technology*. 1999. London, United Kindom.

11. Bernier, Y.W. *Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization*. in *Proc. of Game Developers Conference'01*. 2001.

12. Mauve, M., J. Vogel, V. Hilt, and W. Effelsberg, *Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications.* IEEE Transaction on Multimedia, 2004. 6(1): p. 47-57.

13. Beigbeder, T., R. Coughlan, C. Lusher, and J. Plunkett. *The Effects of Loss and Latency on User Performance in Unreal Tournament 2003*. in *SIGCOMM'04 Workshops*. 2004.

14. Nicholes, J. and M. Claypool. *The Effects of Latency of Online Madden NFL Football*. 2004.

15. Sheldon, N., E. Girard, S. Borg, M. Claypool, and E. Agu. *The Effect of Latency on User Performance in Warcraft III*. in *NetGames*. 2003.

16. Quax, P. and P. Monsieurs. *Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game*. in *SIGCOMM'04 Workshops*. 2004.

17. Eric Cornin, B.F., Anthony R. Kurc, Gugih Jamin, *An Efficient Synchronization Mechanism for Mirrored Game Architecture*, in *Multimedia Tools and Applications*. 2002. p. 7-30.

18. Funkhouser, T.A. *RING: A Client-Server System for Multiuser Virtual Environments*. in *In Proc. 1995 Symposium on Interactive 3D Graphics*. 1995.

19. Roccetti, M., S. Ferretti, and C.E. Palazzi. *The Brave New World of Multiplayer Online Games: Synchronization Issues with Smart Solutions*. in *11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*. 2008.

20. Lin, Y.-J., K. Guo, and S. Paul. *Snyc-Ms: Synchronized Messaging Service for Real-Time Multi-Player Distributed Games*. in *Proceedings of the 10th IEEE Conference on Network Protocols*. 2002.

21. Lety, E., L. Gautier, and C. Diot. *MiMaze, a 3D Multi-Player Game on the Internet*. in *Proceeding of the 4th International Conference on Virtual System and MultiMedia*. Nov 1998. Gigu, Japen.

22. Ahmed Abdelkhalek, A.B. *Parallel and Performance of Interactive Multiplayer Game Servers*. in *Proceedings of the 18th Parallel and Distributed Processing Symposium*. 2004.

23. Pantel, L. and L.C. Wolf. *On the Suitability of Dead Reckoning Schemes for Games*. in *NetGames*. 2002. Germany: ACM.

24. Mauve, M. *How to keep a dead man from shooting*. in *Proc. of the 7th International Workshop on Interactive Distributed Multimedia Systems*. 2000.

25. Steinman, J.S. *Interactive SPEEDS*. in *Proceedings of the 24th annual symposium on Simulation*. 1991. New Orleans, Louisiana, United States: IEEE Computer Society Press.

26. Steinman, J.S. *Breathing Time Warp*. in *Proceedings of the seventh workshop on Parallel and Distributed Simulation*. May 1993.

27. Armitage, G.J. *An Experimental Estimation of Latency Sensitivity In Multiplayer Quake 3*. in *11th International Conference on Networks*. 2003. Sydney.

28. Varvello, M., F. Picconi, C. Diot, and E. Biersack. *Is there life in Second Life?* in *International Conference On Emerging Networking Experiments And Technologie*. 2008. Madrid, Spain.

29. Abdelkhalek, A., A. Bilas, and A. Moshovos. *Behavior and Performance of Interactive Multi-Player Game Servers*. in *2001 IEEE International Symposium on Performance Analysis of System and Software*. 2003.

30. E.Cronin, B. Filstrup, and A. Kurc, *A Distributed MultiPlayer Game Server System*: EECS589, Coruse Project Report, University of Michigan, May2001.

31. Mauve, M., S. Fischer, and J. Widmer. *A Generic Proxy System for Networked Computer Games*. in *Netgames 2002*. 2002. Braunschweig, Germany: ACM.

32. Pellegrino, J.D. and C. Dovrolis. *Bandwidth Requirement and state consistency in three multiplayer game architecture*. in *NetGames 2003*. 2003. Redwood City, California: ACM Press.

33. Barrus, J.W., R.C. Waters, and D.B. Anderson, *Locales: Supporting Large Multiuser Virtual Environments.* IEEE Computer Graphics and Applications, 1996. 16(6): p. 50-57.

34. Unknown-Author, *Overview of ButterFly Grid*, http://www.emergentgametech.com.

35. Boustead, P. and F. Safaei. *Comparison of Delievery Architecture for Immersive Audio in Crowded Network Games*. in *NOSSDAV'04*. 2004. Cork, Ireland: ACM.

36. Bauer, D., S. Rooney, and P. Scotton. *Network Infrastructure for Massively Distributed Games*. in *NetGames 2002*. 2002. Braunschweig, Germany: ACM.

37. Ritter, J., *Why Gnutella Can't Scale*. 2001, http://www.darkridge.com/~jpr5/doc/gnutella.html.

38. Howe, A.J., *Napster and Gnutella: a Comparison of Two Popular Peer-to-Peer Protocols*. 2001, http://members.tripod.com/ahowe_ca/pdf/napstergnutella.pdf.

39. Hagsand, O., *Interactive multiuser VEs in the DIVE system.* Multimedia, 1996. 3(1): p. 30-39.

40. Boukerche, A., R.B. Araujo, and M. Laffranchi. *A Hybrid Solution to Support Multiuser 3D Virtual Simulation Environments in Peer-to-Peer Networks*. in

Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications. 2004.

41. Rooney, S., D. Bauer, and R. Deydier, *A Federated Peer-to-Peer Network Game Architecture.* IEEE Communications Magzine, 2004. 42(5): p. 114-122.

42. Webb, S.D. and S. Soh. *Cheating in networked computer games – A review*. in *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*. 2007. Perth, WA, Australia.

43. Bharambe, A., J.R. Douceur, J.R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang. *Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games*. in *SIGCOMM*. 2008. Seattle, Washington, USA.

44. Tarng, P.-Y., K.-T. Chen, and P. Huang. *An analysis of WoW players' game hours*. in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*. 2008. Worcester, MA, USA.

45. Ratnasamy, S., P. Francis, M. Handley, R. Karp, and S. Shenker. *A Scalable Content-Addressable Network*. in *Proceedings of SIGCOMM 2001*. 2001.

46. Stoica, I., R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. *Chord: A scalable peer-to-peer lookup service for internet applications*. in *Proceedings of SIGCOMM 2001*. 2001.

47. Rowstron, A. and P. Druschel. *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*. in *Proc. IFIP/ACMMiddleware*. 2001. Heidelberg, Germany.

48. Knutsson, B., H. Lu, W. Xu, and B. Hopkins. *Peer-to-peer support for massively multiplayer games*. in *INFOCOM*. 2004.

49. Kulkarni, S. *Badumna Network Suite: A decentralized network engine for Massively Multiplayer Online applications*. in *IEEE Ninth International Conference on Peer-to-Peer Computing*. 2009.

50. Knussson, B., H. Lu, W. Xu, and B. Hopkins. *Peer-to-Peer Support for Massively Multiplayer Games*. in *INFOCOM*. 2004.

51. Castro, M., P. Druschel, A.M. Kermarree, and A.I.T. Rowstron, *Scribe: A Large-Scale and Decentralized Applicatioin-Level Multicast Infrastructure.* IEEE Journal on Selected Areas in Communications, 2002. 20(8).

52. Kim, J., D. Chang, T. Kwon, Y. Choi, and E. Yuk. *Traffic Characteristics of a Massively Multi-player Online Role Playing Game*. in *NetGames'05*. 2005. New York, USA.

53. Morse, K.L., *Interest Management in Large-Scale Distributed Simulations*, in *Technical Report ICS-TR-96-27*. 1996, University of California: Ivine.

54. Jiang, X., F. Safaei, and P. Boustead. *Latency and Scalability: A Survey of Issues and Techniques for Supporting Networked Games*. in *International Conference on Networks*. 2005. Malaysia.

55. Ratnasamy, S., M. Handley, R. Karp, and S. Shenker. *Topologically-Aware overlay construction and server selection*. in *INFOCOM*. 2002.

56. Jiang, X., F. Safaei, and P. Boustead. *Latency and Scalability: A Survey of Issues and Techniques for Supporting Networked Games*. in *7th IEEE International Conference on Networks (ICON 2005)*. 2005. Kuala Lumpur, Malaysia.

57. *www.bigworldtech.com*.

58. Zhang, X.Y., Q. Zhang, Z. Zhang, G. Song, and W. Zhu, *A Construction of Locality-Aware Overlay Network: mOverlay and Its Performance.* IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, 2004. 22(1): p. 18-28.

59. Ferreira, R.A., S. Jagannathan, and A. Grama. *Enhancing Locality in Structured Peer-to-Peer Networks*. in *Tenth International Conference on Parallel and Distributed Systems (ICPADS'04)*. 2004.

60. Zegura, E., K. Calvert, and S. Bhattacharjee. *How to Model an Internet*. in *IEEE INFOCOM*. 1996.

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

62. Ratnasamy, S., M. Handley, R. Karp, and S. Shenker. *Application-level multicast using content-addressable networks*. in *Proceedings of NGC*. Nov. 2001.

63. Zhao, B., J. Kubiatowicz, and A. Joseph, *Tapestry: An infrastructure for fault-resilient wide-area location and routing*. April 2001, U.C.Berkeley, Tech.Rep. UCB//CSD-01-1141, April 2001.

64. Castro, M., M.B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. *An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer Overlays*. in *INFOCOM*. 2003.

65. S., Z., B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz. *Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination*. in *NOSSDAV*. June 2001.

66. Zhao, B.Y., Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz. *Brocade: Landmark Routing on Overlay Networks*. in *1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*. 2002.

67. Xu, Z., M. Mahalingam, and M. Karlsson. *Turning Heterogeneity into an Advantage in Overlay Routing*. in *INFOCOM*. 2003.

68. Pittman, D. and C. GauthierDickey. *A measurement study of virtual populations in massively multiplayer online games*. in *the 6th ACM SIGCOMM workshop on Network and system support for games*. 2007. Melbourne, Australia.

69. , D. Stutzbach, and R. Rejaie. *Understanding churn in peer-to-peer networks*. in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. 2006. Rio de Janeriro, Brazil.

70. Kim, J., D. Chang, and T. Kwon. *Traffic Characteristics of a Massively Multi-player Online Role Playing Game*. in *Netgames'05*. 2005. New York, USA.

71. Ta, D.N.B. and S. Zhou, *A two-phase approach to interactivity enhancement for large-scale distributed virtual environments.* Computer Networks, 2007. 51(14): p. 4131-4152.

72. Lui, J.C.S. and M.F. Chan, *An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems.* IEEE Transactions on Parallel and Distributed Systems, 2002. 13(3): p. 193-211.

73. Rhea, S., D. Geels, T. Roscoe, and J. Kubiatowicz. *Handling Churn in a DHT*. in *USENIX Annual Technical Conference, ACTE'04*. June 2004.

74. Hoang, G.N., H.N. Chan, K.N. Van, T.L.t. Xuan, and T.N. Manh. *Performance improvement of Chord Distributed Hash Table under high churn rate*. in *International Conference on Advanced Technologies for Communications*. 2009. Hai Phong.

75. Li, J., J. Stribling, T.M. Gil, R. Morris, and M.F. Kaashoek. *Comparing the performance of distributed hash tables under churn*. in *IPTPS*. 2004.

76. Oua, Z., E. Harjulaa, O. Kassinena, and M. Ylianttila, *Performance evaluation of a Kademlia-based communication-oriented P2P system under churn.* Computer Networks, 2010. 54(5): p. 689-705.

77. Ou, Z., E. Harjula, and M. Ylianttila. *Effects of different churn models on the performance of structured peer-to-peer networks*. in *IEEE 20th International*

Symposium on Personal, Indoor and Mobile Radio Communications. 2009. Tokyo.

78. Gummadi, K., R. Gummadiy, S. Gribblez, S. Ratnasamyx, S. Shenker, and I. Stoica. *The impact of DHT routing geometry on resilience and proximity*. in *ACM SIGCOMM*. 2003.

Appendix A: Pastry Overlay Settings for Simulation

Data Structure

The data structure of a node in the overlay used in the simulation is modeled in Matlab. An example of the data structure of a node in the overlay is given in **Fig. A.1.1** as follows.

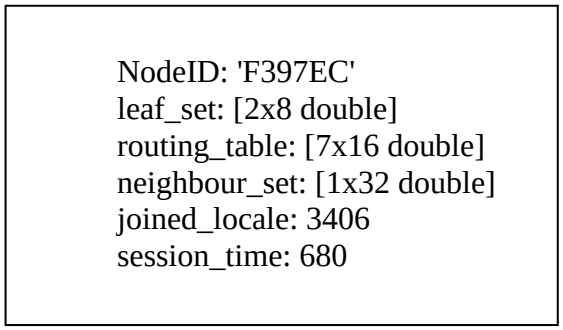


Figure A.1: A Node data structure example

leaf_set

In a ring based structured P2P overlay such as Pastry, The *leaf_set* L is the set of nodes with the |L|/2 numerically closest larger NodeIds, and the |L|/2 nodes with numerically closest smaller NodeIds, relative to the present node's NodeId. For instance, given the present NodeID = F397EC as in **Fig. A.1.1**, the leaf_set of the node is shown in Table A as follows.

F37E5B	F377EC	F37BB9	F37E5B	F384A8	F387CB	F38EE4	F3905E
F3C4C6	F3C920	F3CFA4	F3DD59	F3E5D1	F40719	F40ACA	F40CB1

Table A. 1 The leaf_set of a Sample Node

routing_table

In Pastry, the routing scheme is based on the prefix matching of the NodeId. An example of the routing table for the node in **Fig.A.1.1** is shown in **Table A.1.2** as follows.

Row 1	0CE73C	1AEE74	26020B	39EDAE	47F9BF	5CA46E	656368	7B9F44
Row 2	F0880C	F1F6AE	F266C8		F4204F	F5F9DD	F6A5AD	F7A1B5
Row 3	F30A2D	F3105C	F3274D	F332F0	F349C0			F377EC
Row 4	F3905E							

Row 1	8EF3A8	99586F	AECC38	BF2224	C712CD	D6C25B	E1883B	
Row 2	F821F3	F9F287	FAE884	FB9E2B	FC1DBC	FDC539	FEC668	FF255D
Row 3	F38EE4				F3C920	F3DD59	F3E5D1	
Row 4								

Table A. 2: The routing_table of a sample node

Note that, for the sample NodeID = F397EC, the entries in the first row of its routing_table have no matching prefix with F397EC. As the row number increases, the number of matching prefix increases. For instance, in row 2 of the routing_table, every entry has NodeID starting with digit F, which matches the first digit of F397EC; in row 3 of the routing table, every entry has NodeID starting with digit F3, which matches the first and second digit of F397EC.

route_recursively.m

A routing path in Pastry follows a digital prefix-matching scheme. For instance, to route from node_1 with NodeID = F397EC, to node_300 with NodeID = A58621, the routing path is: F397EC → AECC38 → A5FC5D → A58621, note each hop matches one more digital prefix with the destination node.

A recursive function is written in Matlab to facilitate the routing in Pastry simulation, and the source code is shown in **Fig. A.2** as follows.

```
function path = route_recursive(source,dest,node,record)
%a function find a routing path recursively.
%e.g., route_recursive(1,5,olay,[1 5])

if(isinleafset(source,dest,node) == 0 && isinrtable(source,dest,node) == 0)
   if(source == dest)
      disp('source equals dest');
      path = [source];
      return
   end
   id1 = char(node(source).NodeID);
   id2 = char(node(dest).NodeID);
   match = prefixMatch_v2(id1,id2);
   row = match+1;
   col = hex2dec(id2(match+1))+1;
   source = node(source).rtable(row,col);
   if(source == 0)  %disp('lost path');
      record = [record(1:length(record)-1),-1,record(length(record))];
      path = record;
      return
   end
   record = [record(1:length(record)-1),source,record(length(record))];
   path = route_recursive(source,dest,node,record);
else
   path = record;
   return
end

function flag = isinleafset(n1,n2,node)
%function checks if n2 is in n1's leafset
%return 1 if  true,0 if false
flag = ismember(n2,node(n1).leafset);

function flag = isinrtable(n1,n2,node)
flag = ismember(n2,node(n1).rtable);
```

Figure A.2 Source code for finding a routing path in Pastry

Appendix B: GT-ITM Internet Topology

Transit-Stub Hierarchical Topology

We use GT-ITM topology generator to generate a two-level hierarchical topology, with each top level node representing a transit domain, and with the second level giving each intra-domain graph. A Transit-Stub graph is illustrated in Figure B.1 as follows.

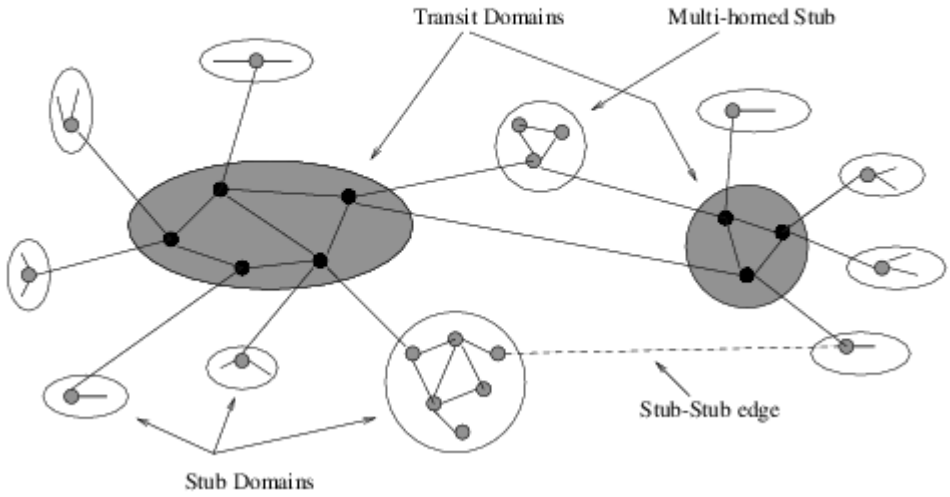


Figure B.1 Transit-Stub Topology [61]

To use the topology generator to generate a graph, an input file with the topology parameters is required. An example of the input file is given in Figure B.2 as follows.

```
# 3 stub domains per transit, no extra transit-stub or stub-stub edges
# 1 transit domain, fully connected
# transit domains have average of four nodes, edge prob 0.6
# stub domains have average of 8 nodes, edge prob 0.4
3 0 0
1 10 3 1.0
4 20 3 0.6
8 10 3 0.4
```

Figure B.2 An input file for the topology generator

The topology generated by the input file in Figure B.2 can be viewed in **Fig B.3** as follows.

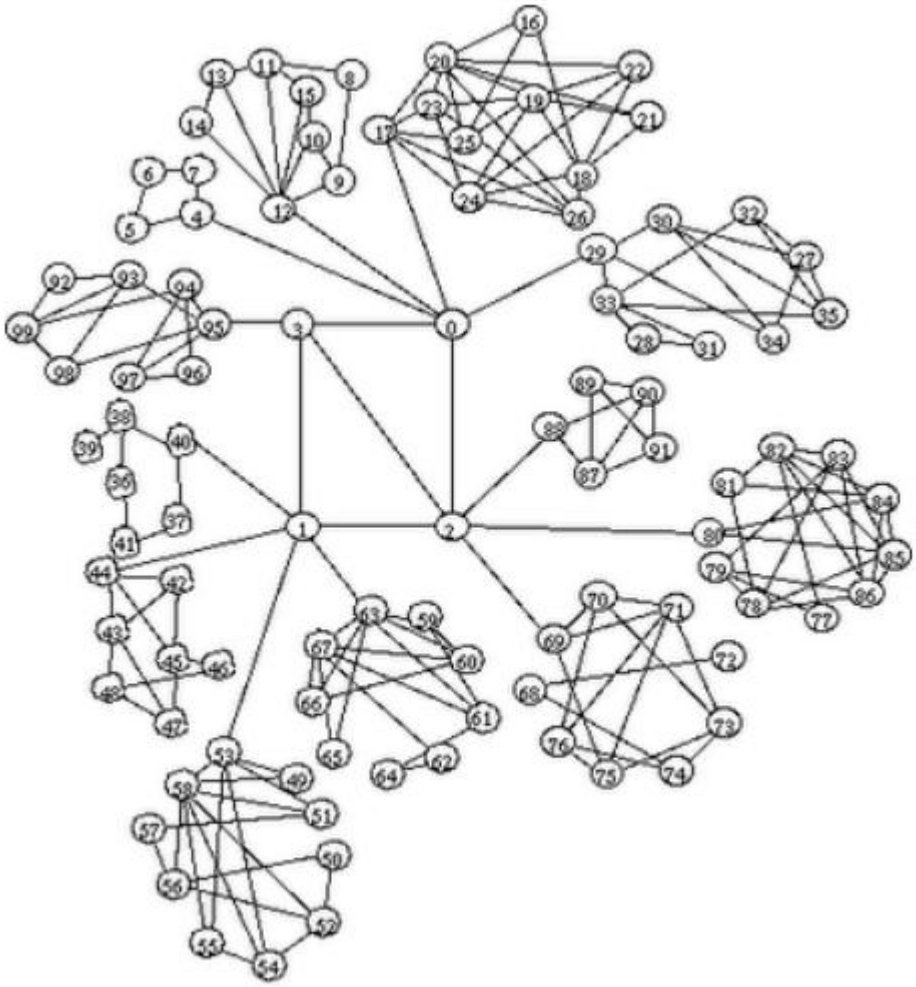


Figure B.3 A topology generated by the example input file

Topology Settings for Thesis Simulation

An input file example for the simulation in this thesis is illustrated in **Fig B.4** as follows.

```
# <method keyword> <number of graphs> [<initial seed>]
# <# stubs/trans node> <#rand. t-s edges> <#rand. s-s edges>
# <n> <scale> <edgemethod> <alpha> [<beta>] [<gamma>]
# number of nodes = 3x8(1+3x8) = 600
ts 10 47
3 0 0
3 20 3 1.0
8 20 3 0.5
8 10 3 0.5
```

Figure B.4 An example of topology input file for the thesis simulation

The input file in **Fig. B.4** indicates to the topology generator to generate a Transit-Stub topology graph with three transit domains, and each transit domain will have, on average, eight nodes. Furthermore, for each transit nodes, there will be eight stub domain created, with eight stub nodes on average in each stub domain. For detailed parameter settings, pleaese refer to [61].