A network-centric approach to interactivity enhancement for large-scale distributed virtual environments

Ta Nguyen Binh Duong

2006

Ta, N. B. D. (2006). Network-centric approach to interactivity enhancement for large-scale distributed virtual environments. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/35753>

<https://doi.org/10.32657/10356/35753>

0455737

**A Network-Centric Approach to
Interactivity Enhancement for Large-Scale
Distributed Virtual Environments**

Ta Nguyen Binh Duong

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2006

Acknowledgements

First of all, I would like to express my deep gratitude to Asst/Prof. Suiping Zhou, my supervisor, for his inspiring instructions and constant supports during this research. I acknowledge his patience and wisdom, which have guided me throughout the past years in graduate school with lots of challenges and difficulties. I am grateful to my mother and my younger sister for their love. I also would like to thank all my friends and the lab technicians in the Parallel & Distributed Computing Centre (PDCC) for their supports.

Last, but definitely not least, thank to Tram Anh, for being with me during the most difficult times. Without your help and understanding I would not be able to finish this thesis. Love you.

Ta Nguyen Binh Duong
Singapore, Jan 2006

I would like to thank the internal and external reviewers for their insightful and constructive comments that help to improve the contents and presentation of my thesis.

Ta Nguyen Binh Duong
Singapore, May 2006

I would like to thank the members of the Oral Examination Panel, A/Prof. Lee Bu Sung, A/Prof. Cai Wentong, Asst/Prof. Tang Xueyan, Asst/Prof. Shen Haifeng, for their efforts and time in my oral defence on 15 Sep 2006.

Ta Nguyen Binh Duong
Singapore, Oct 2006

Table of Contents

Acknowledgements	i
Table of Contents	iii
Abstract	vii
List of Tables	ix
List of Figures	xi
Abbreviations	xv
1 Introduction	1
1.1 Background	2
1.1.1 The concept of DVEs	2
1.1.2 The developments of DVEs	3
1.2 Motivation of this study	8
1.2.1 The challenges in developing DVEs	9
1.2.2 Important research issues	13
1.3 Primary objective of this study	14
1.4 The approach of this study	15
1.5 Summary of contributions	17
1.6 Thesis organization	18
2 DVE architecture and partitioning	21
2.1 Selecting a suitable architecture	21
2.1.1 Peer-to-peer architecture	22
2.1.2 Client-server architecture	24
2.1.3 Multiple-server architecture	25
2.1.4 Other hybrid architectures	28
2.1.5 Discussion	29
2.2 Selecting a partitioning approach	30
2.2.1 Mirrored approach	31
2.2.2 Effect of inter-server communications on scalability	33
2.2.3 Zone-based approach	37
2.2.4 Discussion	42

2.3	Summary	43
3	Server placement approaches	45
3.1	Server placement problem	46
3.1.1	The Web replica placement problem	46
3.1.2	Server placement problem in DVEs	48
3.2	Server placement approaches for DVEs	49
3.2.1	Random placement	50
3.2.2	Degree-based placements	50
3.3	Summary	53
4	Client assignment problem	55
4.1	Notations and concepts	56
4.2	The client assignment problem (CAP)	59
4.3	A two-phase approach to the CAP	60
4.3.1	The initial assignment problem (IAP)	61
4.3.2	The refined assignment problem (RAP)	63
4.4	Related work	64
4.4.1	Client-to-Server assignment in DVEs	64
4.4.2	Classical combinatorial optimization problems	66
4.5	Summary	68
5	Client assignment algorithms	69
5.1	Algorithms for the IAP	70
5.1.1	Random assignment of zones (RanZ)	70
5.1.2	Greedy assignment of zones - algorithm 1 (GreZ-1)	70
5.1.3	Greedy assignment of zones - algorithm 2 (GreZ-2)	74
5.2	Algorithms for the RAP	74
5.2.1	“Normal” inter-server networks	74
5.2.2	Well-provisioned inter-server networks	75
5.3	Two-phase algorithms for the CAP	77
5.4	Optimal algorithm using branch-and-bound	78
5.5	Practical considerations	79
5.5.1	Obtaining input data	79
5.5.2	Dynamic executions of assignment algorithms	83
5.6	Client assignment versus server placement	83
5.7	Summary	84
6	Evaluation methodology	85
6.1	Network models	85
6.1.1	Network topologies	85
6.1.2	Network delay model	91
6.1.3	Shortest-path routing	91
6.1.4	AS-level hierarchical routing	91

6.2	Workload models	92
6.2.1	Client distributions	92
6.2.2	Physical world-virtual world correlation	93
6.2.3	Bandwidth usage	94
6.3	Performance measures	95
6.4	Summary	95
7	Results	97
7.1	Default settings and notations	97
7.2	Impacts of server placement approaches	98
7.2.1	Impacts of the number of servers	101
7.2.2	Impacts of server capacity	101
7.2.3	Impacts of correlations	102
7.2.4	Impacts of client distributions	103
7.2.5	Summary	104
7.3	Impacts of client assignment algorithms	105
7.3.1	“Normal” inter-server networks	105
7.3.2	Well-provisioned inter-server networks	115
7.3.3	Summary	124
7.4	Combined impacts of server placement and client assignment	125
7.4.1	Tight delay bound	126
7.4.2	Bad server placement	127
7.5	Summary	128
8	Conclusions and future directions	131
8.1	Contributions	132
8.2	Future directions	135
	Bibliography	137
A	Author’s publications	151
A.1	Journal papers	151
A.2	Conference papers	152

Abstract

In recent years, advances in networking technologies, computer graphic and CPU power have enabled the rapid development of Distributed Virtual Environments (DVEs). In essence, large-scale DVEs refer to a class of applications that enable thousands of geographically distributed users to simultaneously interact with each other in the same synthetic, computer-generated 3D virtual world. Prominent applications of DVEs include online games, military simulations, collaborative designs, virtual shopping mall, etc. Especially, the latest study by the market research company IDC (<http://www.idc.com>) on the online gaming market of Asia/Pacific (excluding Japan) revealed that the subscription revenue is about US\$1.09 billion in 2004, an increase at about 30% compared to 2003. Also according to this study, this market will be more than double in 2009.

Since DVEs are human-in-the-loop applications, their success greatly depends on the users' perceptions while interacting in the virtual world. Thus, enhancing the interactivity of DVEs is of crucial importance, since the users may not feel that they are *interacting* with other elements in the virtual world if the responses from the DVE are much slower than what the users experience in real-life. Hence, in this thesis, our primary goal is to develop efficient mechanisms to enhance the interactivity of DVEs.

In general, seeking a good balance between enhancing interactivity and other important issues in DVEs such as maintaining consistency, managing resource bottlenecks

and securing the virtual world system under various technical problems such as large Internet latency, high resource demands and high potential of security threats is a very challenging task. In this thesis, we have proposed a *network-centric* approach, which is based on the multiple-server networking architecture and the zone-based, discrete virtual world partitioning approach, to enhance the interactivity of DVEs by *directly* minimizing the network latencies in client-server communications. In this approach, we propose to consider two key problems: the *server placement problem* and the *client assignment problem*. Both problems are NP-hard. To this end, we have developed and adapted several *degree-based* server placement approaches and some *two-phase* client assignment algorithms. Extensive simulation studies with realistic models have shown that our approach is very effective in enhancing the interactivity of large-scale DVEs.

List of Tables

1.1	Internet round-trip latencies measured by the tool King [56]	9
1.2	Bandwidth limitations. Source: http://en.wikipedia.org .	11
6.1	Network topologies used in the simulations	90
6.2	Distribution types	92
7.1	<i>pQoS</i> with imperfect input data, F-BA topology, normal inter-server networks	113
7.2	<i>pQoS</i> with imperfect input data, H-BA/W topology, normal inter-server networks	115
7.3	Impacts of client assignments on <i>pQoS</i> , well-provisioned inter-server networks, small configuration	115
7.4	Impacts of client assignments on resource utilization, well-provisioned inter-server networks, small configuration	116
7.5	Impacts of client assignments on <i>pQoS</i> , well-provisioned inter-server networks, large configuration	116
7.6	Impacts of client assignments on resource utilization, well-provisioned inter-server networks, large configuration	117

7.7 *pQoS* with imperfect input data, H-BA/W topology, well-provisioned inter-server networks 122

7.8 Resource utilization with imperfect input data, H-BA/W topology, well-provisioned inter-server networks 122

List of Figures

1.1	Asia/Pacific online gaming subscription revenue in 2004. Source: [64]	4
1.2	Total active subscribers over time. Source: http://www.mmmogchart.com	5
1.3	Screenshots of some currently popular MMOGs	5
1.4	Military training using DVEs	6
1.5	Screenshots of two popular academic DVEs: PARADISE and DIVE	8
2.1	Common networking architectures	22
2.2	Multiple-server architectures	25
2.3	Mirrored partitioning approach	32
2.4	Multiple-server based DVE using HLA RTI	34
2.5	Screenshots of the prey-predator federation, with 2 servers	35
2.6	Effects of inter-server communication	36
2.7	Seamless partitioning	38
2.8	Seamless partitioning: proxy and shadow objects	39
2.9	Discrete partitioning with “doors” and “ways” connecting different partitions	41
3.1	Max-router placement approach	51
4.1	Geographically distributed server architecture (GDSA)	56

5.1 IAP - Random assignment of zones 71

5.2 IAP - Greedy assignment of zones - algorithm 1 71

5.3 RAP - Greedy assignment of clients 76

6.1 BRITE topology visualized by Otter 86

7.1 Impacts of server placement - CDF with flat topologies 99

7.2 Impacts of server placement - CDF with hierarchical topologies 100

7.3 Impacts of the number of servers on server placement approaches 101

7.4 Impacts of server capacity on server placement approaches 102

7.5 Impacts of virtual world-physical world correlations on server placement
approaches 102

7.6 Impacts of client distributions on server placement approaches - $pQoS$. . 103

7.7 Impacts of client distributions on server placement approaches - resource
utilization 103

7.8 Impacts of client assignment - CDF with flat topologies, small configura-
tion, normal inter-server networks 106

7.9 Impacts of client assignment - CDF with hierarchical topologies, small
configuration, normal inter-server networks 107

7.10 Impacts of client assignment - CDF with hierarchical topologies, large
configuration, normal inter-server networks 107

7.11 Impacts of client assignment - CDF with flat topologies, large configura-
tion, normal inter-server networks 108

7.12 Impacts of the number of servers on client assignment algorithms, normal inter-server networks 109

7.13 Impacts of server capacity on client assignment algorithms, normal inter-server networks 109

7.14 Impacts of physical - virtual world correlation on client assignment algorithms, normal inter-server networks 110

7.15 Impacts of client distributions on client assignment algorithms - *pQoS*, normal inter-server networks 111

7.16 Impacts of client distributions on client assignment algorithms - resource utilization, normal inter-server networks 111

7.17 Impacts of imperfect inputs - F-BA topology, normal inter-server networks 113

7.18 Impacts of imperfect inputs - H-BA/W topology, normal inter-server networks 114

7.19 Impacts of client assignment algorithms - CDF, well-provisioned inter-server networks, small configuration 116

7.20 Impacts of client assignment algorithms - CDF, well-provisioned inter-server networks, large configuration 118

7.21 Impacts of the number of servers on client assignment algorithms - H-BA/W topology, well-provisioned inter-server networks 119

7.22 Impacts of server capacity on client assignment algorithms - H-BA/W topology, well-provisioned inter-server networks 119

7.23 Impacts of physical world-virtual world correlations on client assignment algorithms - H-BA/W topology, well-provisioned inter-server networks . . 120

7.24 Impacts of client distributions on client assignment algorithms - H-BA/W topology, well-provisioned inter-server networks 120

7.25 Impacts of inter-server network delays on client assignment algorithms - H-BA/W topology 121

7.26 Impacts of imperfect inputs - H-BA/W topology, well-provisioned inter-server networks 123

7.27 Combined impact - tight delay bound, flat topologies 126

7.28 Combined impact - tight delay bound, hierarchical topologies 127

7.29 Combined impact - bad server placement, flat topologies 128

7.30 Combined impact - bad server placement, hierarchical topologies 129

Abbreviations

AS Autonomous System

CAP Client Assignment Problem

CDN Content Distribution Network

CloC Closest-server Assignment of Clients

DIS Distributed Interactive Simulation

DVE Distributed Virtual Environment

FPS First Person Shooter Games

GAP Generalized Assignment Problem

GDSA Geographically Distributed Server Architecture

GNP Global Network Positioning

GreC Greedy Assignment of Clients

GreZ-1 Greedy Assignment of Zones, algorithm 1

GreZ-2 Greedy Assignment of Zones, algorithm 2

HLA High Level Architecture

IAP Initial Assignment Problem

LDSA Locally Distributed Server Architecture

MILP Mixed Integer Linear Programming

MMOGs Massively Multi-player Online Games

MPGs Multi-player Games

SPGs Single-player Games

RAP Refined Assignment Problem

RTI Runtime Infrastructure

RanZ Random Assignment of Zones

VirC Virtual Location based Assignment of Clients

Chapter 1

Introduction

In recent years, advances in high-speed networking technologies, computer graphics and CPU processing power have enabled the rapid development of *Distributed Virtual Environments* (DVEs). Nowadays, DVE systems can be seen in many areas, such as collaborative design [46], military simulations [79], e-learning [88] and multi-player games [101]. DVEs focus on *real-time* interactions among multiple users in a shared, computer-generated 3D virtual world, even though the users may be distributed at different geographic locations. In the academic community as well as the industry, much research has been conducted on different aspects of DVE systems. However, how to efficiently enhance the interactivity of DVEs is still an open question that needs more research. In this thesis, our goals are to investigate different networking architectures and to develop efficient algorithms to enhance the interactivity of large-scale DVEs.

In this chapter, we introduce the background of this research. We start with a definition of DVEs and the essential properties of DVEs. Then, we review the historical developments of DVEs, as well as some well-known examples of DVEs, including online games, military simulations, etc. Last, we discuss the challenges in developing DVEs,

CHAPTER 1. INTRODUCTION

related research issues, the primary objective and the approach of this study. Finally, we end this chapter with a summary of this thesis's contributions and organization.

1.1 Background

1.1.1 The concept of DVEs

In essence, DVEs are distributed systems that allow multiple geographically distributed users¹ to explore and interact with each other in real-time within a shared, computer-generated virtual world with realistic 3D graphics and sound [99], where each client is represented by an *avatar*. A client controls the behavior of his/her avatar by various *actions*, and the *updates* of an avatar's state need to be sent to other clients in the same *zone* of the virtual world to support the real-time interactions among clients. According to [99], a DVE is distinguished by the following common features.

- **A shared sense of space:** DVE users have the illusion of being located at the same place in the virtual world, e.g., the same room, battlefield, etc.
- **A shared sense of presence:** Each DVE user can “see” other users' avatars in the shared space.
- **A shared sense of time:** DVE users can see other users' behaviors in the shared space *as they occurs*.
- **A way to communicate:** DVE users can communicate with each other by gesture, text or voice.
- **A way to share:** DVE users can interact, e.g., shoot, pick up objects, etc. not only with each other but also with the virtual world.

¹In this thesis, the terms “user”, “participant” and “client” all refer to DVE users and are used interchangeably.

CHAPTER 1. INTRODUCTION

1.1.2 The developments of DVEs

In this section, we review the historical developments of DVEs over the past decades as well as current trends in DVEs. In particular, we focus on the developments of DVEs in three sectors, namely the industry sector, the government sector and the academic sector.

Developments of DVEs in the industry sector

The most notable class of DVEs developed in the industry is *Multi-Player Games* (MPGs). From the first appearance of computer games in the early 1960s², as the personal computers became more and more powerful, the popularity of computer games grew quickly. Most computer games in the early time were *Single Player Games* (SPGs), in which a single human player plays against the computer. Later, the fast development of networking technologies brought multi-player capabilities to single player games, which obviously added more fun to game playing since in MPGs, players play against other human players rather than against the computer. At the beginning, most of the MPGs were deployed only on local area networks (LAN), which are referred to as LAN-based MPGs. Good examples of commercial LAN-based MPGs in the early time includes *Doom* released in 1993 and *Quake* in 1996, both by idsoftware [63]. These games, which are commonly referred to as First Person Shooter (FPS) games, typically involve at most a few tens of players trying to shoot at each other or cooperating to kill monsters in a small 3D virtual world, i.e., a small room or building.

Recently, due to the explosive growth of the Internet, the gaming market's focuses

²The first computer game *Spacewar!* was designed for the PDP-1 computers by MIT students [21]

CHAPTER 1. INTRODUCTION

Figure 1.1: Asia/Pacific online gaming subscription revenue in 2004. Source: [64]

have been transferred from small-scale MPGs with a few tens of players playing in “death-match” sessions to large-scale, persistent, evolving virtual worlds. In these games, which are usually referred to as *Massively Multiplayer Online Games* (MMOGs), there are hundreds or thousands of players interacting with each other at the same time. MMOGs have gained significant attentions as a new stream of the entertainment industry. This online gaming market are growing very fast, mainly based on a monthly subscription model: unlike traditional single player games or LAN-based MPGs, MMOG players have to pay a subscription fee each month in order to continue playing the game. The latest study by the market research company IDC (<http://www.idc.com>) on the online gaming market of Asia/Pacific (excluding Japan) revealed that the subscription revenue is about US\$1.09 billion in 2004, with a growing rate approximately 30% compared to 2003 [64] (see Fig. 1.1 for a detailed revenue distribution). Also according to that study, this market will be more than double in 2009. Fig. 1.2 shows the growing number of MMOG subscriber over the past few years.

Some of the most popular MMOGs currently are *Everquest* [11], *Lineage* [83], *World of Warcraft* [36], etc. (screenshots are shown in Fig. 1.3).

CHAPTER 1. INTRODUCTION

Figure 1.2: Total active subscribers over time. Source: <http://www.mmmogchart.com>

(a) Lineage

(b) Everquest

(c) World of Warcraft

Figure 1.3: Screenshots of some currently popular MMOGs

CHAPTER 1. INTRODUCTION

(a) Tanks

(b) Helicopters

Figure 1.4: Military training using DVEs

Developments of DVEs in the government sector

The Department of Defense (DoD) of the United States was one of the pioneers to develop DVEs, for the purpose of military training. The first DVE system, called SIMNET (simulator networking) [79], which was started in 1983 and finished in 1990, is a low-cost solution to train military units using peer-to-peer networked human-in-the-loop simulators. SIMNET creates a virtual environment in which soldiers can enter through networked simulators and interact with each other in the synthetic battlefield. The SIMNET protocols later became the Distributed Interactive Simulations (DIS) protocols [28], which was adopted as the IEEE Standard 1278 in 1992.

In 1995, the DoD developed the *High Level Architecture* (HLA) as a generic framework for distributed simulations, which defines a software architecture for constructing large-scale simulations from different components and facilitating the reuse of simulation components [71]. The HLA specification became the IEEE Standard 1516 in 2000. As a generic software architecture, HLA can be applied in other kinds of distributed interactive simulations outside the military domain.

CHAPTER 1. INTRODUCTION

Developments of DVEs in the academic sector

In parallel with the development of DVEs in the entertainment industry and the DoD, the academic community has conducted research on DVEs for a long time. Some of the pioneering works on DVEs appeared in the academic literature are discussed as follows.

The NPSNET project [37] at the Naval Postgraduate School (NPS), USA, aimed to develop virtual environment technologies that can be useful for the DoD, and can inter-operate with DIS-compliant systems. In NPSNET, a participant can be a human, a vehicle, etc. It is the first virtual environment that can work over the Internet, and is widely used in the DoD as well as university laboratories. The first version of NPSNET was released in 1990, and currently the latest version is NPSNET-V released in 2002.

The PARADISE (Performance ARchitecture for Advanced Distributed Interactive Simulation Environments) project [26], which was initiated in 1993 at Stanford University, USA, is a similar attempt to NPSNET, which aimed to develop large-scale networked simulation environment that can support multi-player interactive, 3D-simulations running over wide-area networks. The latest version is 0.1, which includes a 3D graphic engine and allows multiple simulated objects to move around and see each other in the same virtual environment.

Another well-known DVE project is DIVE (Distributed Interactive Virtual Environment) [23] at the Swedish Institute of Computer Science, Sweden. DIVE is an Internet-based DVE in which participants navigate in 3D space and see, meet and interact with other users. The first DIVE version was released in 1991. DIVE is well-integrated with the World Wide Web (WWW). It can also read and export Virtual Reality Modelling Language (VRML) and several other 3D formats.

CHAPTER 1. INTRODUCTION

(a) PARADISE

(b) DIVE

Figure 1.5: Screenshots of two popular academic DVEs: PARADISE and DIVE

Summary

Currently, it seems that there is a convergence of the industry, government and academic community in research and development of DVEs. For instance, recently some commercial and online gaming systems have been developed using the DoD HLA standard, e.g., the OpenSkies MMOG framework [18]. On the other hand, the US Army Force [4] are currently using online games to train military skills for their soldiers. In addition, more and more academics have been conducting research in online games, especially MMOGs. There are also some academic conferences devoted to online gaming issues, for example, the NetGames conference series [16]. All of these things promise that we would be witnessing much more impressive developments of DVEs in the near future.

1.2 Motivation of this study

Although DVEs are very popular nowadays, there are still many challenging research issues that need to be solved before we can realize their full potential. In this section, we discuss the challenges in developing DVEs as the main motivations of this study. We then summarize various important research issues in DVEs.

CHAPTER 1. INTRODUCTION

1.2.1 The challenges in developing DVEs

Internet latency

Table 1.1: Internet round-trip latencies measured by the tool King [56]

Source	Target	Latency
210.245.0.18 (Vietnam)	155.69.5.156 (Singapore)	170ms
18.7.22.83 (USA)	155.69.5.156 (Singapore)	231ms
18.7.22.83 (USA)	202.108.22.30 (China)	430ms
155.69.5.156 (Singapore)	137.194.2.17 (France)	304ms
18.7.22.83 (USA)	137.194.2.17 (France)	90ms
18.7.22.83 (USA)	131.170.40.30 (Australia)	234ms
202.183.214.46 (Thailand)	155.69.5.156 (Singapore)	30ms

As the Internet is used as the communication medium for DVEs, users located around the world can interact with each other in the same virtual world easily. However, the Internet latency, one of the biggest problems with the current Internet architecture, is a very challenging issue that DVE developers must face. Typically, small-scale LAN-based DVEs are not suffered from this problem since the message transmission delays in LANs are negligible. The situation is different in the Internet. In [41], the author pointed out that it is the latency in the Internet that is hindering the development of real-time, interactive multi-user applications like DVEs. Due to the limit of the speed of light, the round-trip latency of a signal sent over fiber optics from a point to another point on the opposite side of the Earth is about 200ms, given that the speed of light in glass fibre is about 200000km/s. Moreover, network congestions, processing overheads at the operating systems and network hardware such as routers, modems and network cards will make this latency even larger. Table 1.1 lists some round-trip network latency values that we measured using the tool King [56].

CHAPTER 1. INTRODUCTION

The large Internet latency is the major cause of several difficult challenges for DVE developers. Being a multi-user, interactive application, two of the most critical issues in DVEs are *consistency* and *interactivity*. The consistency of a DVE can be defined as the similarity in the DVE state at every users participating in the application [100]. For example, when user *A* moves, user *B* must also see *A*'s movement on his computer screen. An *absolute consistency* must guarantee that each user has the same view of the DVE state at the same time. In the above example, absolute consistency requires that user *B* must see the same movement of *A* (distance, direction, velocity, acceleration, etc.) at the same time on his screen as that on user *A*'s screen. In practice, absolute consistency is not achievable. Due to large network latency, inconsistent views among users may seriously damage the users' experiences in interacting within the DVE.

Another critical requirement for DVEs is the interactivity. More specifically, the interactivity of a DVE characterizes how fast a human user can see or feel the result of his/her actions compared to what happens in real life [110]. When the interactivity is bad, i.e., the delay of the result is large, the user may have difficulties in interacting with the DVE (controlling avatar, manipulating objects, etc.). With the presence of large network latency, there is a difficult trade-off between consistency and interactivity in DVEs. For example, when a user *A* moves, for consistency reasons, we need to ensure that this information is successfully propagated to user *B* in the same virtual world before it can be displayed on *A*'s screen. Hence, if the time for transmitting messages from *A* to *B* is large, i.e., the network latency between *A* and *B* is large, the interactivity of the DVE may be damaged due to the large delay in displaying the result of *A*'s action. On the other hand, if we choose to display the result of *A*'s action immediately, the consistency

CHAPTER 1. INTRODUCTION

of the DVE could not be guaranteed.

In practice, depending on the type of DVEs, different latency criteria are required to ensure a good trade-off between consistency and interactivity. For example, Multi-player Real-Time Strategy (RTS) games like Microsoft's *Age of Empires* [13] typically require a latency of $500ms$ [101], while FPS games and car-racing games have much more stringent latency requirements, about $250ms$ [58] and $100ms$ [91], respectively.

Resource demands

Table 1.2: Bandwidth limitations. Source: <http://en.wikipedia.org>.

Connection	Bandwidth limitation
Modem	56Kbps
ISDN	64 or 128Kbps
ADSL	64Kbps-1024Kbps (upstream), 256Kbps-8Mbps (downstream)
Satellite Internet	64Kbps-1Mbps (upstream), 128Kbps-16Mbps (downstream)
T1 (T-carrier)	1.54Mbps
E3 (E-carrier)	34.368Mbps
Frame Relay	8Kbps-45Mbps
Ethernet	10-1000Mbps
OC3 (Optical carrier)	155.52Mbps

LAN-based DVEs are typically limited to a small number of users. The scenario is different in large-scale DVEs, where a large number of users (on the order of thousands) around the world may interact with each other in the same virtual world simultaneously. Various types of resources may be consumed in a DVE: network bandwidth, CPU cycle, memory, etc. Table 1.2 lists the bandwidth resource limitations of different types of network connections. In addition, interactions among distributed users could be very resource-consuming in terms of CPU cycles, for example, collision detection, agreement

CHAPTER 1. INTRODUCTION

and resolution among participants [99]. In a peer-to-peer DVE, in which users communicate directly with each other, the network and the computer of each user have to handle these resource demands. On the other hand, in a client-server DVE, in which DVE users connect to one or more servers, it is the servers' responsibility to provide the users with necessary resources. For DVE developers, how to distribute and adapt a limited amount of resources to support as many users as possible is a challenging task.

Security vulnerabilities

Since DVEs are deployed over the Internet, they would also suffer from security vulnerabilities similar to those in all Internet-based systems [67]. For example, private and important information about users, such as credit card information (used to pay subscription fees in MMOGs for instance), username and password to join the DVE may be stolen by various common hacking techniques. Currently, online game servers are also the favorite target of Distributed Denial of Services (DDoS) attacks, which may lead to the loss of revenue for the game hosting company due to the fact that players may get frustrated and decide to leave the game after they could not connect to unavailable game servers.

In addition, since DVEs like online games are human-in-the-loop applications, cheating is very common and is very difficult to prevent. Cheating players may modify client softwares to gain unfair advantages such as immortal and invisible capabilities. They can even earn real money from selling items obtained by cheating in the virtual world. This problem is known in the popular MMOG Everquest [11]. Cheating is considered a very serious problem in online games today, since it discourages fair players to continue playing the games. Thus, the ability to provide a fair playing environment for all players

CHAPTER 1. INTRODUCTION

is critical to the success of online games.

1.2.2 Important research issues

To support large-scale, real-time, interactive applications like DVEs, various research issues need to be investigated. In this section, we briefly summarize the most important research issues in DVEs.

Networking architectures

Networking architectures define how participants in the DVE can communicate with each other. There are some common architectures [99] such as peer-to-peer, client-server, multi-server architecture, etc. Choosing a suitable communication architecture is very important since it provides the basis for developing other techniques to deal with various challenges in DVE developments.

Enhancing interactivity and consistency

Consistency and interactivity are some of the most critical requirements for DVEs. How to improve interactivity and consistency of DVEs is closely related to the networking architecture used. Various mechanisms have been studied in the literature, e.g., dead reckoning [72] which is used to hide the effects of network delay by predicting the states of remote users, or the local lag approach [76, 78, 107] which aims to strike a good balance between consistency and interactivity by using an appropriate “playout delay” [94] for each local user’s actions.

CHAPTER 1. INTRODUCTION

Resource management

Large-scale DVEs require efficient resource management. Various types of resource can be consumed in a DVE, including network bandwidth, CPU processing cycle, server memory, etc. Common resource management techniques in DVEs are as follows. Interest management [30, 55, 75, 82, 105, 112] is used to reduce the number of exchanged messages in the DVE by sending messages only to the users that need them. Load distribution and DVE partitioning [38, 61, 80, 84, 104] is used to avoid bottlenecks in multiple-server architectures when there are many users gathering on a server. Resource scheduling [39, 40] is employed to provide a graceful degradation of system's performance when there are many objects in the system competing for the limited resource.

Security

Neglecting the security and cheating issues in DVEs like online games may result in serious troubles for the success of the games. To protect DVEs from security vulnerabilities and cheating, techniques such as encryption, DDoS attack preventions, asynchronous synchronization protocol, etc. are commonly employed [67, 68, 109].

1.3 Primary objective of this study

So far, we have discussed the technical challenges as well as important research issues in developing large-scale DVEs. In our view, the interactivity is the most important requirement for DVEs, since it is the most likely factor that reflects the *usability* of a DVE. What we mean by usability is as follows. One may spend significant efforts to develop a DVE with highly realistic 3D graphic, sound, etc., but those features may

CHAPTER 1. INTRODUCTION

become meaningless if the interactivity of that DVE is poor, i.e., the users do not feel like they are interacting with other elements in the virtual world if the responses from the DVE are much slower than what the users experience in real life. Having realized the significance of the interactivity requirement in the success of a DVE, the primary objective of this study is to *investigate efficient mechanisms to enhance the interactivity of DVEs*. However, at the same time, we should not neglect other important issues, e.g., consistency control, resource management and security threats.

1.4 The approach of this study

Seeking a good balance between enhancing interactivity and other important issues such as maintaining consistency, managing resource bottlenecks and securing the virtual world system is a very challenging task. To enhance interactivity, traditional *application-centric* approaches, e.g, dead reckoning, usually aim to “hide” the effect of network latency by predicting the state of remote users. Since the activities of human users in DVEs are highly unpredictable, such approach may result in visual perception errors, or in more serious cases, inconsistent DVE states. In this thesis, we propose a *network-centric* approach for enhancing DVE’s interactivity via *directly reducing network latencies in communications between DVE’s elements*. The outline of our approach is as follows.

- First, an appropriate networking architecture for DVEs is needed. In this study, we select a multiple-server networking architecture, which essentially provides good ways to implement consistency control, resource management and security guarantees. Second,

CHAPTER 1. INTRODUCTION

in order to handle large-scale DVEs, it is essential to employ a “divide-and-conquer” approach, i.e., to partition the large virtual world into multiple smaller worlds, i.e., zones.³ In this thesis, we investigate various partitioning approaches, and select a *zone-based, discrete* partitioning approach, in which the virtual world is partitioned into multiple distinct zones. This partitioning approach is the currently most popular one in commercial DVEs due to its simplicity and efficiency.

- We then propose to enhance interactivity via exploiting servers’ and user’s network locations. In particular, we aim to answer the following two key questions:

- *Where to place the servers to minimize the network latencies in communications between users and servers?*

- *How to assign geographically distributed users to servers, again, to minimize the network latencies in communications between users and servers?*

The first question, referred to as the *server placement* problem, to some extent is similar to the Web replica placement problem [93], which is a fundamental problem in Content Distribution Networks (CDNs). This problem consists of determining good network locations to place the replicas of a Web site to minimize network latencies for its users. Various replica placement mechanisms for CDNs have been developed over the past few years [44, 90, 93]. However, to the best of our knowledge, there’s no existing work that evaluates whether these mechanisms can still be applied well to the case of DVEs. Hence, our task in this study is to adapt existing placement mechanisms from CDNs, and to evaluate their appropriateness in DVE’s context.

The second question involves assigning each client that is interacting in the virtual

³In this thesis, the terms “zone” and “partition” are used interchangeably to refer to a part of the virtual world.

CHAPTER 1. INTRODUCTION

world to the most appropriate server. Unlike CDNs where each client can just simply connect to its closest replica, the assignment problem in DVE is much more complicated. To this end, we need to consider both the network location and the virtual world location (in short, virtual location) of each client. The virtual location of a client can be determined by the client's current zone of the virtual world in which its avatar is currently residing. To this end, we formulate a new problem, termed the *client assignment* problem, in which the ultimate goal is to seek efficient assignments of clients to servers to enhance the interactivity of DVEs. This problem is shown to be NP-hard. Hence, efficient heuristic solutions are required.

1.5 Summary of contributions

Here, we summarize the main contributions of this thesis with regard to the primary objective discussed above. For more details, please refer to Section 8.1.

- We have proposed a network-centric methodology for enhancing DVE interactivity via *directly reducing network latencies in communications between DVE's elements*.
- We have reviewed and evaluated several *networking architectures* for DVEs.
- We have discussed and evaluated several *partitioning approaches* for DVEs.
- We have proposed to consider the *server placement* problem to reduce the client-server communication delays in DVEs. We then have adapted and proposed several server placement approaches for DVEs.
- We have proposed and formulated a new problem, termed the *client assignment* problem. A two-phase approach, which consists of an initial assignment and a refined assignment, is proposed to solve this problem. We then have developed several simple

CHAPTER 1. INTRODUCTION

yet effective heuristic algorithms for this problem.

- We have built a very realistic simulation model for evaluating the proposed approaches and algorithms.
- We have conducted extensive simulations to evaluate the effectiveness of the proposed server placement approaches and client assignment algorithms.

In addition, over the past few years, I have published and submitted several refereed papers in international conferences and journals. For the full list of papers, please refer to Appendix A.

1.6 Thesis organization

The thesis is organized as follows.

- Chapter 1 introduces the background of DVE research and the historical developments of DVEs. We then discuss the motivation, primary objective, and approach of this study. A short summary of the contributions is also given.
- Chapter 2 discusses various networking architectures and partitioning approaches for DVEs. We analyze several aspects of DVE architectures and partitioning approaches, including scalability, consistency maintenance, etc. We then select the most appropriate architecture and partitioning approach.
- Chapter 3 describes the replica placement problem in Content Distribution Networks, and our server placement problem in DVEs. We then adapt and analyze several server placement approaches that are applicable to DVEs.
- Chapter 4 formulates and analyzes a new problem, termed the *client assignment*

CHAPTER 1. INTRODUCTION

problem. In this chapter, we also describe our proposed *two-phase* approach which consists of an *initial assignment* phase and a *refined assignment* phase to the client assignment problem. Both phases are proved to be NP-hard.

- Chapter 5 presents and analyzes several algorithms for the client assignment problem. We also discuss some practical considerations in implementing these client assignment algorithms, and the relations between server placement and client assignment.
- Chapter 6 presents our evaluation methodology for the proposed assignment algorithms and server placement approaches.
- Chapter 7 presents simulation results to evaluate server placement approaches and client assignment algorithms. We first study the individual impact of server placement and client assignment, then consider their combined impacts on the interactivity of DVEs.
- Chapter 8 summarizes the achievements of this thesis and suggests some interesting future directions.

Chapter 2

DVE architecture and partitioning

As mentioned in the previous chapter, our approach to enhancing DVE’s interactivity is based on the multiple-server architecture and the zone-based, discrete partitioning approach. Networking architecture is very important since it defines how various components of a DVE can communicate with each other. On the other hand, to efficiently handle large-scale DVEs, it is essential to employ a “divide-and-conquer” approach to partition the huge virtual world into smaller worlds. In this chapter, we review several well-known, widely used DVE architectures and partitioning approaches. We assess each architecture and partitioning approach in details to justify our decision, i.e., selecting the multiple-server architecture and the zone-based, discrete partitioning approach.

2.1 Selecting a suitable architecture

In this section, we discuss several popular DVE architectures, namely peer-to-peer, client-server, multiple-server, and some hybrid architectures.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

Figure 2.1: Common networking architectures

2.1.1 Peer-to-peer architecture

In the peer-to-peer (P2P) architecture, as shown in Fig. 2.1(a), there is no central server, and all clients (users) are connected to each other in a peer-to-peer manner. Each client calculates its own view of the DVE state, and distributes updates generated from its actions to all other clients interacting in the same virtual world. Due to the unavailability of multicast communication in the current Internet, message dissemination among the clients in P2P architecture is usually done using point-to-point communications. This architecture has the following advantages.

- Communication latency is reduced due to the fact that messages are sent from one client to another directly.
- There is no single point of failure, due to the fully distributed nature of the architecture, the DVE will not break down due to the abrupt failure of one client.
- The architecture is cost-effective, since each client contributes its resource (CPU power, network bandwidth) to the system automatically.

However, the P2P architecture exposes multiple drawbacks, which may severely hinder its application. Some of them are listed as follows.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

- The potential of cheating and security problems is high. Since data are located at each client's machine, it becomes easier for one user to modify the data to gain unfair advantages over other user.
- Consistency maintenance is difficult. Since each client keeps its own view on the DVE state and there's no authoritative management of the shared state, a consistent view of the DVE state among clients is very hard to achieve due to the large network latency in the Internet. Inconsistent views among users in the same DVE may greatly damage the users' experience.
- The scalability of the system is limited by the capacity of the weakest client. For instance, a client with low-bandwidth connection will not be able to handle the network traffic generated by a large number of clients in the same DVE. Thus the maximum number of clients in that DVE may be very limited.

Despite the attractiveness of a low-cost solution, to the best of our knowledge there's currently no commercial large-scale DVE that employs P2P architecture. Instead, it is mainly implemented in some research systems, which are usually limited in functionalities and features. Well-known examples includes the game *MiMaze* [47] developed by C. Diot and L. Gautier in 1999, and the NPSNET system [37], developed at the Naval Postgraduate School, USA. Recently, B. Knutsson et al [69] have developed a large-scale DVE (in fact a simple MMOG prototype) using Distributed Hash Table (DHT) in P2P networks. Their experiments showed that this prototype system can support thousands of simultaneous simulated users,¹ however security and consistency issues are still open problems in that work.

¹A user in this experiment is not a real human user, instead it is a computer program capable of simulating typical user's activities.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

In other examples, military simulation systems like SIMNET [79] also use P2P architecture, but the participants in such system are known in advance and are interconnected via high-speed, private networks. Some P2P RTS and FPS games, for example id software's *Doom* [63] and Microsoft's *Ages of Empires* [13] are also popular, but the number of simultaneous participants in these games are very limited, i.e., on the order of a few tens.

2.1.2 Client-server architecture

In the client-server (CS) architecture, as shown in Fig. 2.1(b), each client connects to a central server. The server maintains the overall DVE state. Each client will communicate with the server to send its actions and receive updates generated by other clients. All the clients' actions are serialized and processed at the server, then updates are disseminated to all interested clients. The advantages of CS architecture are as follows.

- Cheating prevention could be easier to implement since all application data are stored in the central server. It is rather difficult for the cheating users to illegally modify the data if the server is well-protected, e.g., by using firewalls.
- Consistency maintenance is also straightforward since the server maintains the overall DVE state.

The disadvantages of CS architecture are as follows.

- Communication delay is increased since actions issued by a client have to be sent to the server before updates are distributed to other clients.
- The expenses on server hardware and network bandwidth are typically high.
- A single server introduces the single point of failure problem. It could also be

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

Figure 2.2: Multiple-server architectures

overloaded if the number of clients is large, thus may damage the interactivity of the DVE.

Currently, the CS architecture is mostly used in commercial online games with a limited number of players such as *Quake* [63].

2.1.3 Multiple-server architecture

Generally, it is observed that the advantages of P2P architecture are the disadvantages of CS architecture and vice versa. Thus, the multiple-server (MS) architecture which combines P2P and CS may be able to strike a good balance between these two architectures. In this hybrid architecture, as shown in Fig. 2.2, multiple servers are connected to each other in a peer-to-peer manner usually via high speed networks (e.g., LAN or other kinds of private, dedicated networks), and each client connects to one of these servers. Depending on the distribution of servers, we have two variants of MS architecture, namely *locally distributed server architecture* (LDSA) and *geographically distributed server architecture* (GDSA).

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

Locally distributed server architecture (LDSA)

In the LDSA as shown in Fig. 2.2(a), servers are all placed at a central location, e.g., a machine room. The connections among servers are usually done via LAN technologies, e.g., Gigabit Ethernet. The LDSA retains all advantages of the CS architecture, plus the improved scalability because now load distribution can be implemented to alleviate overloaded servers. However, the single point of failure problem still remains, since the Internet connection for the whole server cluster may fail. Someone may argue that multihoming² can resolve this problem. However, we should note that in multihoming, two independent network links from different ISPs may actually share the same transmission line [15]. This problem effectively forms a single point of failure in the network connections.

In addition, the communication delay between client and server may still be high if the client is far away (in terms of network delay) from the server. For example, in Fig. 2.2(a), client c_3 , which is currently residing in the partition³ z_2 of the virtual world, may have large communication delay to its server s_3 . This may damage the experience of c_3 in interacting with the DVE.

The LDSA has been mostly employed in commercial large-scale online games with *loose* real-time requirements, like Everquest [11]. Recently, the needs for more interactivity in online games have been increased [102], thus better networking architectures need to be investigated.

²Multihoming is a networking technique to enhance the reliability of the Internet connection of an IP network. In this technique, the network in question establishes connections to two or more completely different ISPs [15].

³We will discuss DVE partitioning in details later in this chapter.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

Geographically distributed server architecture (GDSA)

Deploying servers at a central geographic location like the LDSA may result in low scalability and low interactivity for DVEs. Hence, the GDSA [34, 77] aims to overcome these limitations while retaining the advantages of the LDSA. In this architecture, as shown in Fig. 2.2(b), multiple servers are distributed over the Internet. All geographically distributed servers may be fully meshed with well-provisioned network connections to facilitate fast exchanges of data among servers. We note that many existing Content Distribution Networks (CDNs) like Akamai [3], Exodus [12], etc. have adopted such architecture to support the huge access demands to popular Web sites.

To support fast communication among geographically distributed servers in the GDSA, one approach is to deploy a private, high-speed networks to interconnect the servers. However, this may be expensive. A more viable alternative is to opt for an overlay distribution approach [48, 87]. In this approach, the overlay-based server-network service provider purchases network resources, e.g., bandwidth, with certain level of QoS guarantees from individual network domains via *bilateral service level agreement* (SLA). Based on the SLAs with different network domains, the overlay-based service provider can build a *logical* end-to-end service delivery infrastructure on top of existing networks. Each logical connection between any two overlay nodes, referred to as *service gateways*, is provided by the underlying network domain with purchased QoS guarantees. DVE developers may rent servers and network resources from such third-party providers to avoid excessive deployment and management costs [48].

The overlay approach is particularly suitable for geographically distributed architectures like the GDSA. Even in the case that there's no QoS guarantee in the network

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

connections among overlay nodes as mentioned above, the overlay approach can still reduce the loss rate, latency and improve throughput of network communication significantly. This is achieved by closely monitoring the network status of the links between overlay nodes to determine the most suitable routing path. For example, the Resilient Overlay Network (RON) in [31] deploys its node across the Internet. These RON nodes monitor the quality of the Internet paths among themselves, and use this information to decide whether to route packets directly over the Internet or via other RON nodes, hence, application-specific routing metrics can be optimized. RON's experiments have shown that it is able to reduce the network latency on slow Internet paths by *tens to hundreds of milliseconds*, while incurring small overheads.

In the GDSA, with appropriate assignments of clients to servers, the interactivity can be greatly improved. For example, this is the case in Fig. 2.2(b) when client c_3 , which is interacting in partition z_2 , connects to its closest server s_3 , which is currently hosting partition z_2 .

2.1.4 Other hybrid architectures

Recently, there are some new hybrid networking architectures for DVEs have been proposed. Notable examples include the peer-to-peer with *central arbiter* architecture (P2P-CA) [92] and the *federated* peer-to-peer architecture (FP2P) [97]. The P2P-CA architecture is in fact a modified version of the P2P architecture in an effort to enhance the consistency maintenance. In this architecture, each client not only send messages to all other clients, but also to a central arbiter. The central arbiter, which is similar to the server in CS architecture, is used to detect and resolve inconsistencies. Whenever an

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

inconsistency occurs among the clients, the central arbiter sends a corrected message to rollback the DVE states of all involved clients. This architecture still suffers from scalability issues since the central arbiter needs to simulate the entire DVE state in order to detect inconsistencies. Additionally, the possibility of cheating is also high since data are located at each client's machine, similar to the P2P architecture.

The FP2P architecture improves P2P architecture in a different way from P2P-CA by focusing on scalability issues. This architecture proposes the use of multiple *multicast reflectors*, which in essence are fast application-aware packet forwarders to disseminate messages between clients interacting in the same virtual world. Clients send messages to the multicast reflectors which in turn distribute these messages to all other interested clients. To do this, each multicast reflector maintains a network address list of clients that are interacting with each other. Multicast reflectors remove the scalability bottlenecks caused by limited bandwidth and processing power at each client in the P2P architecture. However, consistency and cheating are still open issues in FP2P architecture, since each client still calculates its own view of the DVE state.

2.1.5 Discussion

Different networking architectures for DVEs have different pros and cons, and the selection of an architecture could raise other research issues that need to be investigated. For instance, P2P architecture can improve the interactivity while raising the critical needs for better scalability, consistency maintenance and anti-cheating mechanisms, which may be very difficult to achieve.

On the other hand, server-based architectures have the advantages of straightforward

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

consistency maintenance, security protection and improvable scalability. The most challenging problem in server-based architectures is how to enhance the interactivity in light of large Internet latency. The LDSA, due to its centralization property, may provide few opportunities to address this challenge, i.e., we can only consider to place the server cluster at a good location in the network to reduce the communication delay. However, as we will show later in this thesis, placing servers at a desirable location is not always achievable, due to several technical and economic constraints.

Hence, we believe that the GDSA is a better choice than the LDSA in enhancing the interactivity of DVEs. As discussed above, the GDSA has all the advantages of a server-based architecture, plus the fact that it is more scalable than the LDSA, and the possibility to enhance the interactivity via finding good assignment of clients to well-placed geographically distributed servers. Hence, in this thesis, we select the GDSA for further study. In the following, for convenience, we shall refer to the GDSA as the multiple-server architecture, unless we explicitly specify that we are using the LDSA. Our primary objective is to investigate efficient server placement and client-to-server assignment mechanisms to enhance the interactivity of GDSA-based DVEs.

2.2 Selecting a partitioning approach

In this section, we discuss and evaluate two common DVE partitioning approaches, namely the mirrored approach and the zone-based approach. In addition, in the zone-based approach, we further classify it into seamless partitioning and discrete partitioning.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

2.2.1 Mirrored approach

In the mirrored partitioning approach [42, 43], the whole virtual world is *replicated* at every server in the system. More specifically, each server has a complete copy of the virtual world, and all servers in the system need to exchange data frequently to synchronize the content of the virtual world. For this synchronization purpose, each server needs to send *all* information received from its clients, i.e., the clients that connect directly to it, to *all* other servers in the system. Fig. 2.3 illustrates a possible scenario in a DVE with mirrored partitioning approach. In this figure, although server 1 has only one connected client (its avatar is represented by a black dot), it has to maintain 3 copies of the avatars belonging to server 2 (represented by 3 circles). These copies are referred to as *shadow objects*, while the original avatars are referred to as *proxy objects* [62]. In total, server 1 has to maintain 4 avatar objects: 1 proxy and 3 shadow. Similarly, server 2 maintains 3 proxy and 1 shadow object.

As a result of data replication, one of the most attractive properties of the mirrored approach is that a client can connect to any server in the system to explore the virtual world. Hence, in a geographically distributed server architecture (i.e., GDSA), each client can choose to connect to the closest mirror to reduce the network latency to the server. In practice, this can be done by utilizing an Internet distance service such as IDMaps [53]. Once a client-server connection is done, the client does not need to switch to another server when it moves around in the virtual world. In the zone-based partitioning approach which will be mentioned later in this chapter, clients need to change server when they move from zone to zone.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

Figure 2.3: Mirrored partitioning approach

This partitioning approach shares some similarities with the web replication mechanism [93] in well-known CDNs like Akamai [3], Exodus [12], etc. to support the huge access demands to popular Web sites. However, the mirrored partitioning approach has several weaknesses that are currently preventing it from wide deployments. First, unlike the replication of web documents, DVE replication faces serious consistency issues [111] which may damage the users' experience in interacting with the virtual world. Maintaining a consistent state among distributed copies of the virtual world under a heterogeneous networking environment like the Internet is a very challenging task. So far, consistency maintenance for DVEs has been investigated only for small, trivial virtual worlds, e.g., in the simple game MiMaze [47], or under very restricted environments, e.g., in [43]. For large-scale DVEs with many servers and thousands of geographically distributed users interacting concurrently in real-time, good consistency maintenance for mirrored partitioning approach is unlikely to be achieved for practical use.

Second, for a large-scale virtual world with hundreds or thousands of concurrent users, replication is very resource-consuming in terms of CPU cycle and network bandwidth

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

due to inter-server communications.⁴ Thus, the mirrored approach is not scalable. In the following, we conduct some simple experiments to demonstrate this important problem.

2.2.2 Effect of inter-server communications on scalability

In order to investigate the effect of inter-server communication on the scalability of the mirrored approach, we have implemented a simple DVE system with a multiple-server architecture⁵ based on the HLA specification [71]. The HLA was developed in 1995 and became the IEEE standard 1516 in 2000 as a generic framework to promote the reusability and interoperability of components in large scale distributed simulations. In the HLA, a distributed simulation is called a *federation*, and each simulator in the simulation is referred to as a *federate*. The HLA Interface Specification describes several services to support distributed simulations: Federation Management, Object Management, Declaration Management, Time Management, etc. In the implementation, we use the HLA Run-Time Infrastructure (RTI), which is an implementation of the HLA Interface Specification, to facilitate communications among servers. The overall architecture of our prototype is shown in Fig. 2.4. Each server is implemented as an HLA federate, and all the servers form an HLA federation.

Prototype implementation

The prototype is called the *prey-predator federation*, in which two types of clients (each represented by an avatar), namely the prey and the predator, move around and interact in a 2D virtual world. Ideally, in the experiment, each prey or predator should be under the control of a human user. However, due to the lack of a large number of human

⁴Inter-server communication refers to the exchange of data among servers in the DVE.

⁵In this case, the LDSA is used, since we are only interested in the scalability of the mirrored approach, not the client-server network delay.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

Figure 2.4: Multiple-server based DVE using HLA RTI

participants, we decided to simulate the human users as autonomous agents (implemented in Java). Each autonomous agent will control one prey or predator. In the 2D virtual world, the prey moves around in a flocking manner [95], i.e., they tend to move in groups. Preys tend to flee from the predators when the predators come close. The predator also moves around in the environment. When it sees a prey that is close to it, it will pursue the prey until it is close enough to the prey, i.e., the prey is within its interaction range, and then kills the prey. To keep the number of preys in the federation constant, a new prey is born whenever a prey is killed.

The code for server and client execution is similar to the code structure of a very popular multi-player client-server game, Quake [29], and is implemented in Java. Communications between clients and servers are done via non-blocking sockets using Java NIO [59]. The servers in our DVE system are glued together to form a federation by utilizing DMSO RTI1.3NG-V6 [10]. HLA services used in the federation include Federation Management, Object Management and Declaration Management services.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

Figure 2.5: Screenshots of the prey-predator federation, with 2 servers

Experiments and results

The servers used in the experiments are all Dell Workstation PWS350 (Pentium IV 2.8 GHz, 512MB RAM) machines. The client program runs on multiple Dell Optiplex GX260 (Pentium IV 2.4 GHz, 512MB RAM) machines. Up to 50 Java threads (50 clients) are allocated to each client machine. This is acceptable, since we observed that CPU utilizations of the client machines are low, at about 10%-15%. All machines in the federation run Windows 2000 Professional, and are connected to each other via 100Mbps Ethernet, thus the network latency in the experiments is negligible.⁶ One of the Dell Workstations is dedicated to run the RTI Executive process (rtiexec) and the Federation Executive process (fedexec). To study the effect of inter-server communication, in all experiments, we keep the server workload balanced by assigning the same number of clients to each server. Some screenshots of the federation are shown in Fig. 2.5. The red objects are predators, and the black ones are preys.

⁶Our aim here is to study the effect of inter-server messages on system scalability only. Network latencies among servers may cause serious inconsistencies, but that effect has been discussed in the first chapter, and hence not quantitatively studied here.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

As shown in Fig. 2.6(a) (the solid line, with inter-server communication), adding more servers to the system definitely improves the average response time (ART).⁷ However, the improvement seems not very significant. On the average, when one more server is added, the ART decreases by about 8.2%. This could be due to a highly increased volume of inter-server communications in the mirrored approach, as shown in Fig. 2.6(b), since every message received at a server must be propagated to all other servers in the system. These inter-server messages may take considerable amount of CPU cycles to send, receive and update (i.e., to reflect the new states of the shadow objects) in each server.

Figure 2.6: Effects of inter-server communication

To further investigate the effect of inter-server communication on the response time,

⁷The response time for a client is measured from the time the client sends a message to its server to the time it receives the server's reply.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

we setup the prey-predator federation in 2 different cases as follows, using a system with 2 servers. The first case is the normal situation, when mutually interested avatars, i.e., avatars that can see and interact with each other, may belong to different servers, which incurs inter-server communication, as shown in Fig. 2.5(a). In the remaining case, we restrict the avatar movements to ensure that the mutually interested avatars are from the same server. As shown in Fig. 2.5(b), there are two *separate* groups of mutually interested avatars, each group belongs to only one server, and the avatars in one group are not aware of any avatar in the other group. Hence, there will be no inter-server communication among servers in the system in this case.

Fig. 2.6(c) shows the ARTs of our DVE system with 2 servers for the two different cases described above. It is obvious that in the case of no inter-server communication, the ART is reduced by about 52.5% on the average compared to the case with inter-server communication. Additionally, as shown in Fig. 2.6(a) (the dotted line), if there's no inter-server communication, the improvement in ART is good when more servers are added to the system. On the average, when one more server is added, the ART decreases by about 34.8%, compared to the improvement of 8.2% when the system has inter-server communication. We conclude that inter-server communication in the mirrored approach may severely limit the scalability of the system, and thus the mirrored approach is not suitable for large-scale DVEs with thousands of simultaneous clients.

2.2.3 Zone-based approach

In the zone-based partitioning approach [11, 74, 104], generally the virtual world is spatially partitioned into multiple *zones*. Each zone constitutes a part of the entire virtual

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

world, e.g., a room in a building, and is handled by only one server. A client can interact with another client in the same zone, or even in adjacent zones. Since each server only needs to handle one or more zones instead of the entire virtual world, this approach is more scalable than the mirrored approach. Depending on whether the zone boundaries are visible inside the DVE, we can further classify the zone-based partitioning approach into seamless partitioning and discrete partitioning. In seamless partitioning, the boundaries are invisible to clients, and a client in a zone can freely interact with another client that belongs to a different zone. In discrete partitioning, two clients must be in the same zone to interact with each other.

Seamless partitioning

Figure 2.7: Seamless partitioning

The seamless partitioning approach is depicted in Fig. 2.7, in which each client is represented by a black dot, and zone may have different shapes, i.e., square, hexagon, etc. In this approach, there is no *visible boundary* between the zones, even though these zones may be hosted on different physical server machines. A client may interact with other clients in other zones, and may travel from zone to zone freely, without knowing that it may be switched from one server to another server during its movement. This may result

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

Figure 2.8: Seamless partitioning: proxy and shadow objects

in a more-enjoyable experience in interacting with the virtual world, although this is not always the case [35]. Seamless partitioning is implemented by letting the servers that hosts adjacent zones communicate with each other. More specifically, a server s needs to send all information received from the clients interacting in its zones to all other servers that are hosting adjacent zones to the zones of s .

Fig. 2.7 shows the seamless partitioning with three different methods to partition the virtual world. The first method is the simplest one using square zones. In this method, each square zone will have at most 8 adjacent zones, as shown in Fig. 2.7. A more sophisticated method for world-partitioning is the *interleaved-squaring* (as illustrated in the middle of Fig. 2.7) proposed in [62]. This method yields a maximum number of 6 adjacent zones for any zone in the virtual world, thus the amount of inter-server communications could be reduced. Finally, hexagonal zones can be used to achieve the similar reduction, with the maximum of 6 adjacent zones for each zone. We note that the latter two may be harder to implement in practice than the simple square method.

The seamless partitioning is more scalable than the mirrored approach, since each server only needs to maintain the shadow objects for clients in adjacent zones which belong to other servers, not for the entire virtual world, as illustrated in Fig. 2.8. In

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

this figure, server 1 needs to maintain in total 5 objects (its 3 proxy objects and 2 shadow objects from server 2), while server 2 maintains 4 objects (its 3 proxy objects and one shadow object from server 1). The number of objects each server needs to maintain is clearly lower compared to the mirrored approach, in which each server has to maintain 6 objects in total. However, this partitioning approach suffers from the same critical drawback as that in the mirrored approach, which is the consistency maintenance issue. As mentioned above, the content of adjacent zones must be synchronized, just like the situation with different copies of the virtual world in the mirrored approach. The only difference between the two approaches lies in the amount of data that need to be synchronized. In this sense, the seamless partitioning requires less synchronization efforts, since it does not maintain the entire copy of the virtual world at every server in the system.

In addition, each client needs to switch server whenever it moves from its current zone to another zone which belongs to a different server. In seamless partitioning, this process of switching server must be done transparently to the human users, otherwise the zone boundaries would become visible. A typical switching process is as follows. When the client enters a zone that belongs to a different server (referred to as the new server), its original server will convert the proxy object representing it to a shadow object. Meanwhile, the new server will convert the shadow object previously representing that client to a proxy object. When the conversions are done, and the client's data are already synchronized at both servers, the client will connect to the new server, and the switching process is complete.

Finally, load distribution for seamless virtual worlds is also complicated. This resource

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

management technique can be implemented by transferring zones from an overloaded server to a less loaded one. However, we need to take into account the inter-server communication incurred by adjacent zones when making zone migration decisions. In fact, this problem is NP-complete [74]. To the best of our knowledge, currently there are few commercial DVEs that employ the seamless partitioning.

Discrete partitioning

Figure 2.9: Discrete partitioning with “doors” and “ways” connecting different partitions

In the discrete partitioning approach, the boundaries between the zones become visible to the clients interacting in the virtual world. In this partitioning approach, two clients must be in the same zone to interact with each other. Fig. 2.9 illustrates the discrete partitioning approach with “doors” and “ways” connecting different partitions. When a client wants to move to another zone, it needs to look for the “door” and the “way” that connect its current zone and its destination. This is natural, though, since we experience similar things in the real world. For example, when we want to move from a room to another room in a big building, we definitely need to find the door to leave the first room and the way that leads to the second room. Since clients in different zones do not interact with each other, the discrete partitioning approach removes the need to maintain

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

the shadow objects for clients that belong to other servers, thus improving scalability and avoiding complex synchronization tasks as required in the seamless approach. The process of switching server is also simpler compared to the seamless approach. When a client leaves its current zone to enter a zone that belongs to a different server, its original server just needs to delete its proxy object. The new server of that client will then create a new proxy object for it, and the client is redirected to its new server.

One more point that favors discrete partitioning is that it offers high flexibility during the design phase of the virtual world, i.e., the zones of the virtual world can be designed independently. Selecting discrete partitioning also relieves the system's computational resource from tracking a client's positions to see whether it goes out the boundary of its zone, since clients can only get out of a zone through the "doors". In a seamless virtual world, clients can move out or in at any feasible point⁸ along the boundary.

With the discrete partitioning approach, load distribution can be implemented in a straightforward manner, compared to the case of seamless partitioning, since there is no inter-server communication among servers, except when a client moves from one zone hosted by one server to another zone hosted by a different server. The discrete partitioning is widely used, especially in large-scale commercial DVEs like MMOGs.

2.2.4 Discussion

We have extensively discussed the pros and cons of various partitioning approaches for DVEs, namely mirrored, seamless and discrete partitioning. The mirrored and seamless

⁸Here, feasible point refers to places in the virtual world that clients can move freely, i.e., without obstacles.

CHAPTER 2. DVE ARCHITECTURE AND PARTITIONING

approaches have critical consistency issues, and their scalability is limited due to inter-server communications, as demonstrated by the experiments. In light of this, the discrete partitioning approach appears to be the most attractive choice to distribute the virtual world over a set of distributed servers.

2.3 Summary

In this chapter, we have discussed several well-known DVE networking architectures and partitioning approaches. Via thorough analysis and experiments, we have selected the multiple-server architecture and the zone-based, discrete (in short, discrete) partitioning approach as the base of our study.

In the following chapters, we will investigate how to enhance the interactivity of DVEs via appropriate server placement approaches and client assignment algorithms.

Chapter 3

Server placement approaches

A key question in the multiple-server architecture is where to place the servers to reduce the communication delay between the clients and the servers. This problem has been studied extensively in the context of Operations Research back in the 1960s as the Facility Location problem [32, 45, 49, 70], and more recently, as the Web replica placement problem [44, 90, 93]. However to our knowledge there's no existing work that assesses the suitability of Web replica placement approaches to the server placement problem in DVEs. In this chapter, we review the state-of-the-art results in the Web replica placement problem. We then discuss the difference between the server placement problem in DVEs and the Web replica placement problem. Finally, we suggest and analyze some possible approaches adapted from existing Web replica placement strategies that we believe appropriate to the server placement problem in DVEs.

CHAPTER 3. SERVER PLACEMENT APPROACHES

3.1 Server placement problem

3.1.1 The Web replica placement problem

The explosive growth of the World Wide Web in the late 1990's had demanded efficient content delivery architectures to offer better service to web users at lower costs. A Content Delivery Network or Content Distribution Network (CDN) is a system of many (hundreds or thousands) well-connected servers deployed over the Internet to transparently deliver web content to end users [9] by allocating the replicas of contents at the edges of the Internet, i.e., close to the users. Hence, in large CDNs like Akamai [3] or Exodus [12], the Web replica placement is crucial to the system performance.

Informally stated, the Web replica placement problem concerns how to choose K web replicas among N potential sites (i.e., network locations) in the network, where $K < N$, to minimize a given objective function, usually the clients's access latencies, or the total bandwidth consumption. After the replicas are in place, each web client connects to its closest replica to retrieve the content needed. Here, it is assumed that each client needs to access only one replica to get all of its interested content. The Web replica placement is usually formulated as a Minimum K-median problem as follows [93].

Definition 3.1.1. Minimum K-median based formulation for the web replica placement problem

Let $N = \{1, \dots, n\}$ be the set of potential sites to place the replicas. Let y_i , $i \in N$ indicate whether the site i is selected to place the replica, with $y_i = 1$ if site i is selected or $y_i = 0$ otherwise. Let d_j denote the access demand at site j and c_{ij} denote the network distance between i and j . $d_j c_{ij}$, $i, j \in N$, is the cost of assigning the set of web clients at

CHAPTER 3. SERVER PLACEMENT APPROACHES

site j to the replica at site i . Let x_{ij} indicate whether the set of web clients at site j is assigned to the replica at site i , with $x_{ij} = 1$ if j is assigned to i or $x_{ij} = 0$ otherwise.

The objective now is to minimize the total cost

$$\sum_{i,j \in N} d_j c_{ij} x_{ij} \quad (3.1.1)$$

subject to

$$\sum_{i \in N} x_{ij} = 1, \forall j \in N, \quad (3.1.2)$$

$$x_{ij} \leq y_i, \forall i, j \in N, \quad (3.1.3)$$

$$\sum_{i \in N} y_i \leq K, \quad (3.1.4)$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in N, \quad (3.1.5)$$

$$y_i \in \{0, 1\}, \forall i \in N. \quad (3.1.6)$$

Remark 3.1.1. The set of constraints (3.1.2) ensures that each set of web client at site j is assigned to a replica site i . The set of constraints (3.1.3) ensures that a replica must be placed at site i before the set of web clients at j can be assigned to i . Constraint (3.1.4) ensures that the maximum number of replica sites is K .

CHAPTER 3. SERVER PLACEMENT APPROACHES

Greedy placement approach

The minimum K-median problem is NP-hard [2]. In [93], the authors proposed a greedy placement algorithm that is shown to be able to perform very well compared to a “super-optimal” algorithm¹ that uses Lagrangian relaxation with subgradient optimization [54]. Via simulations, the median performance of the greedy algorithm is shown to be within a factor of 1.1-1.5 of the optimal.

The basic idea of the greedy placement algorithm is as follows. We have K iterations, at each iteration we select one new site to place the replica. In selecting a replica location, we compute the total cost given in (3.1.1) under the assumption that all web clients connect to their closest replicas, and pick the site that gives the lowest cost in conjunction with the already selected sites. The complexity of this algorithm is $O(N^2K)$.

3.1.2 Server placement problem in DVEs

Although the Web replica placement problem has been well-studied in the context of CDNs with promising results, the appropriateness of the proposed approaches to the server placement problem in DVEs has not yet been studied. The main difference between the Web replica placement problem and the server placement problem in DVEs lies in the “connection” of clients to servers. In the first problem, it’s relatively straightforward to let web clients connect to their closest replicas, hence the greedy placement approach is feasible. However, in DVEs, the large virtual world is partitioned into multiple distinct zones, and each zone is hosted by a separate server. A client cannot simply connect to its closest server since that server may *not* manage its zone. Hence, the greedy placement

¹The “super-optimal” solution may be better than the optimal, but may not be feasible.

CHAPTER 3. SERVER PLACEMENT APPROACHES

approach is not appropriate for the server placement problem in DVEs, as it assumes that clients always connect to their closest servers.

Therefore, for the server placement problem in DVEs, we need an approach that can place the servers in appropriate network locations to reduce clients' communication latencies without knowing *in advance* which server the clients will connect to. In [44, 90], the authors have shown that by placing Web replicas at key network locations, i.e., at the network nodes (routers or Autonomous Systems² with high node degrees,³ the clients' communication latencies are significantly reduced, regardless of clients's locations in the network. We feel that such approaches are applicable to the server placement problem in DVEs. However, the mechanisms in [44, 90] were only evaluated under the assumption that clients would eventually connect to their closest servers, although this assumption was not used in deciding server placements. Hence, in this study, our main task is to adapt the placement approaches in [44, 90] to address the server placement problem in DVEs, and to evaluate whether such approaches are still applicable in the new context. In the following section, we suggest several possible placement approaches. The evaluation results shall be presented in later chapters of this thesis.

3.2 Server placement approaches for DVEs

In this section, we present several server placement approaches which we believe applicable to DVEs. Such approaches include random placement and degree-based placements.

²An autonomous system (AS) in the Internet is a collection of IP networks and routers under the control of one (or sometimes more) entity that presents a common routing policy to the Internet [5].

³In this thesis, the term "node degree" is used to represent the number of links connecting a node with its neighbors [90].

CHAPTER 3. SERVER PLACEMENT APPROACHES

3.2.1 Random placement

The random placement serves as a basic approach for comparison in the sense that any good server placement approach should perform better than the random placement. In this approach, servers are randomly placed in the network.

Remark 3.2.1. The computational cost of the random placement approach is $O(Mm)$, where M is the total number of routers and m is the number of servers.

3.2.2 Degree-based placements

The degree-based server placement approaches aim to place the servers at some key locations in the network. These key locations are usually the network nodes (routers or ASes) with high node degrees. The degree-based approaches have been shown to perform comparably to the greedy server placement algorithm in [93] in the context of the Web replica placement problem [90]. However, whether they are appropriate for DVEs needs to be further investigated. To our knowledge, our work is the first to consider degree-based server placements to enhance the interactivity of DVEs.

Max-router

In the Max-router placement approach, we sort all the routers in the network in descending order of their node degrees⁴ using Quicksort [19], and place servers at the routers with highest node degrees. Intuitively, a node with larger node degree is likely to be closer to other nodes, hence it is a good choice for server placement.

Fig. 3.1 illustrate the Max-router server placement approach. In this figure, router

⁴We determine the node degree of a router, i.e., router-level node degree, by calculating the number of router-to-router links between that router and its neighbors.

CHAPTER 3. SERVER PLACEMENT APPROACHES

Figure 3.1: Max-router placement approach

B has the largest node degree (which is 5). If a server is placed there, then the network distances (in number of hops⁵) from client c_1 and c_2 to the server are 2 and 1, respectively. However, if the server is placed at router A , which has a node degree of 1, then the above client-server distances become 3 and 2, respectively. Thus, placing servers at nodes with large node degree may effectively reduce the client-server communication delay, which implies good interactivity.

Remark 3.2.2. The computational cost of the Max-router placement approach is $O(M \log M) + O(m)$, or $O(M \log M)$, where M is the total number of routers in the network topology and m is the number of servers, $m \ll M$.

Max-AS/Max-router

The Max-AS/Max-router placement approach is similar to the Max-router approach in the sense that they both aim to place servers at nodes with large degree in the network topology. The main difference between the two approaches is that Max-AS/Max-router

⁵Indeed, network hop counts are good indications of round-trip network delays [89].

CHAPTER 3. SERVER PLACEMENT APPROACHES

uses a hierarchical placement approach which is based on the structural nature of real network topologies.

First, Max-AS/Max-router approach selects the ASes with largest AS-level node degrees.⁶ In each AS, we place one server at the router with the largest node degree. By spreading servers over multiple ASes, this approach may further reduce network latencies for geographically distributed clients compared to the Max-router approach.

To implement the Max-AS/Max-router approach, we sort all the routers according to the node degrees of their corresponding ASes (each router belongs to an AS). If there are two routers with the same AS-level node degree, we consider their router-level node degrees. We then iterate over the sorted list of all routers to select one best router in each AS to place the server.

Remark 3.2.3. The computational cost of the Max-AS/Max-router placement approach is $O(M \log M) + O(M)$, or $O(M \log M)$, where M is the total number of routers.

As comparative references, we also consider some alternative server placement approaches, namely Min-router, Max-AS/Min-router and Min-AS/Max-router. The Min-router approach may serve as an “upper bound” on how bad a server placement approach can be. The Max-AS/Min-router and Min-AS/Max-router may help us to identify which one (AS or router) is the major factor in the hierarchical server placement approach. However, in practice, sometimes we have to accept these methods for server placement since typically the routers with high node degrees may be very busy due to the high volumes of network traffic, thus it may be impossible to add any more service to them

⁶We determine the node degree of an AS, i.e., AS-level node degree, by calculating the number of AS-to-AS links between that AS and its neighbors.

CHAPTER 3. SERVER PLACEMENT APPROACHES

[90]. Moreover, economic and administrative constraints may make it harder to place servers at the selected network locations.

Min-router

In this approach, in contrary to the Max-router placement, the routers with minimum node degrees are selected to place the servers.

Max-AS/Min-router

This approach is similar to Max-AS/Max-router, but in each AS, the router with minimum node degree is selected to place the server for that AS.

Min-AS/Max-router

This approach is similar to Max-AS/Max-router, but we select the ASes with minimum node degrees, and in each selected AS we select the router with maximum node degree to place the server.

3.3 Summary

In this chapter, we have described the well-known Web replica placement problem and its most representative research results. We then discuss the difference between the Web replica placement problem and our server placement problem in DVEs. The greedy Web replica placement approach, which has been shown to perform very close to the optimal solution, cannot be applied directly to our problem. Therefore, we need to look for other approaches that can make server placement decisions without knowing in advance the server that each client will connect to. We have suggested and analyzed several

CHAPTER 3. SERVER PLACEMENT APPROACHES

placement approaches for DVEs based on the existing approaches proposed for Web replica placements.

In the next chapter, we shall present another interesting problem that arises in conjunction with the DVE server placement problem. This problem, referred to as the client assignment problem, concerns how to effectively assign clients to servers to further reduce the client-server communication delays. A significant part of this thesis is devoted to formulate, analyze and develop efficient algorithms for the client assignment problem. Unlike server placement approaches which require the ability to place servers at “special” network locations,⁷ client assignment algorithms can be employed in a straightforward manner without any special need of infrastructure. We believe that this problem will strongly complement the server placement problem presented in this chapter. Later in this thesis, we will study the individual impact of server placement and client assignment, as well as the combined impact of these two methods in enhancing the interactivity of DVEs.

⁷Placing a server at a good network location is not always achievable due to technical constraint as already mentioned above, or economic reasons.

Chapter 4

Client assignment problem

In Content Distribution Networks, after the web replica placement is done, assigning clients to servers is relatively straightforward, i.e., each client is connected to its closest server. However, in the context of DVEs with the zone-based, discrete partitioning approach, this simple client assignment approach is not appropriate, since each client may interact in a zone which is not hosted by the client's closest server. In this chapter, we formulate the *client assignment problem* (CAP). Informally stated, the CAP concerns how to assign the clients to the servers in a multiple-server architecture to enhance the interactivity of the DVE. Our formulation in this chapter is applicable for both the LDSA and GDSA, although for the LDSA the CAP becomes trivial, since assigning a client to any server will not affect the network delay from this client to its server.¹ Before formulating the CAP, we first introduce some related definitions and concepts. We then present our *two-phase* approach to the CAP, which consists of an *initial assignment* phase and a *refined assignment* phase. Both phases are proved to be NP-hard. Finally, we discuss some related work to our problem.

¹Recall that in the LDSA, all the servers are placed at the same geographic location, e.g., a machine room.

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

4.1 Notations and concepts

In this study, we focus on DVEs that adopt the geographically distributed server architecture (GDSA) [34, 77] and the discrete partitioning approach, due to their advantages as discussed in the previous chapter. Recall that the GDSA consists of multiple servers geographically distributed over the Internet, as shown in Fig. 4.1. All geographically distributed servers are usually fully meshed with well-provisioned network connections, although this is not a required assumption for our approach.

Figure 4.1: Geographically distributed server architecture (GDSA)

Before formulating the client assignment problem, we introduce the following notations:

- c_i - A client in the DVE.
- $C = \{c_1, \dots, c_k\}$ - The set of all clients in the DVE.
- z_i - A zone in the DVE.
- $Z = \{z_1, \dots, z_n\}$ - The set of all zones in the DVE.
- s_i - A server in the DVE.
- $S = \{s_1, \dots, s_m\}$ - The set of all servers in the DVE.

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

We now introduce the following definitions:

Definition 4.1.1. Contact server

The *contact server* of a client is the server that the client directly connects to. A client only sends inputs to its contact server. The contact server may process the inputs and send updates to the client if it is hosting the client's zone, or it may forward the client's inputs to another server which is hosting the client's zone. We denote the contact server of a client c_j as $s^c(c_j)$.

For example, in Fig. 4.1(a), s_1 is the contact server of c_1 and c_2 .

Definition 4.1.2. Target server

The *target server* of a client is the server that is hosting the client's zone. Inputs from a client will be forwarded to its target server. The target server may send updates to the client directly if it is also the contact server of the client, or it may send to the client indirectly via the client's contact server. All clients in a zone have the same target server (therefore, we may say “the target server of a zone”), while they may have different contact servers. We denote the target server s_i of a client c_j or a zone z_k as $s^t(c_j)$ or $s^t(z_k)$, respectively.

For example, in Fig. 4.1(a), s_1 is both the contact and target server of c_1 and c_2 , i.e., $s^c(c_1) = s^c(c_2) = s^t(c_1) = s^t(c_2) = s_1$. In Fig. 4.1(b), we switch c_2 to server s_2 (but the avatar of c_2 is still in zone z_1), the target server of c_1 and c_2 is still s_1 , the contact server of c_1 is s_1 , while the contact server of c_2 is now s_2 . Inputs from c_2 are forwarded to s_1 by s_2 . Since the network connection between s_1 and s_2 is well-provisioned with little congestion, the communication between c_2 and s_1 may now have shorter delay (assuming

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

s_2 is closer to c_2 than s_1). However, this incurs extra resource utilization for inter-server communication between s_1 and s_2 .

- R_{s_i} - The resource consumption on a server s_i . This can be measured by CPU usage, network bandwidth usage, etc. Since the network bandwidth often represents the major operating cost in current server-based MMOGs [65], in this study, we assume that the server CPU is not a bottleneck, and measure the resource consumption by the network bandwidth usage only.

- $R_{c_i}^T$ - The amount of resource (network bandwidth) utilized by a client c_i on its target server. Note that $R_{c_i}^T > 0, \forall c_i$.

- $R_{c_i}^C$ - The amount of resource utilized by a client c_i on its contact server. Note that $R_{c_i}^C = 0$ if the contact server and target server of c_i are the same, otherwise $R_{c_i}^C = 2R_{c_i}^T$, since all communications between c_i and its target server are forwarded by its contact server (assuming the resource utilization is measured by bandwidth requirement).

- R_{z_i} - The amount of resource utilized by a zone z_i on its target server. We have $R_{z_i} = \sum_{c_j \in z_i} R_{c_j}^T$.

- C_{s_i} - The resource capacity of a server s_i .

- $d_{c_i s_j}$ - The round-trip network delay between a client c_i and a server s_j .

- D - The *delay bound* of a DVE. The delay bound indicates the required upper bound of the round-trip communication delay between a client and its target server to guarantee the interactivity of the DVE. For different types of DVEs, there are different delay bound requirements. For example, Multi-player Real-Time Strategy (RTS) games like Microsoft's *Age of Empires* [13] typically require a latency of 500ms [101], while FPS games and car-racing games have much more stringent latency requirements, about

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

250ms [58] and 100ms [91], respectively. It should be noted that the communication delay between a client and its target server is different from the network delay between the client and its target server. The communication delay is the sum of the network delay and the processing delay at the target server. However, in this study we assume that the server CPU is not a bottleneck, thus the client-server communication delay is determined by the client-server network delay. In the following, the terms “network delay” and “communication delay” are used interchangeably.

For interactive applications like DVEs, the client-server communication delay is the most important *Quality of Service* (QoS) parameter that the system provides to clients [58]. In this study, we say that a client is *with QoS* or *without QoS* if the communication delay between the client and its target server is smaller or larger than the delay bound, respectively.

4.2 The client assignment problem (CAP)

Our client assignment problem (CAP) under consideration is for the static case, i.e., for a snapshot of the system, and the client-to-zone relation at this snapshot is known. With the geographically distributed server architecture and the zone-based partitioning approach, the CAP concerns how to assign the participating clients in a DVE to servers so that the total number of clients with QoS is maximized. We formulate the client assignment problem as follows.

Definition 4.2.1. Client assignment problem (CAP)

For each client c_j in the DVE, given the client-to-zone relation at a snapshot of the system, find the target server s_k and the contact server s_l for c_j at this snapshot to

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

maximize

$$|\{c_j \in z_i : d_{c_j s_k} = (d_{c_j s_l} + d_{s_l s_k}) \leq D, \forall z_i \in Z\}| \quad (4.2.1)$$

subject to

$$R_{s_i} \leq C_{s_i}, \forall s_i \in S \quad (4.2.2)$$

where $|\cdot|$ denotes the cardinality of a set.

Remark 4.2.1. In Definition 4.2.1, if s_l and s_k refer to the same server, then $d_{s_l s_k} = 0$.

Remark 4.2.2. The constraint (4.2.2) ensures that the resource consumption on each server s_i will not exceed its capacity.

4.3 A two-phase approach to the CAP

To address the client assignment problem, in this study we propose a two-phase approach as follows. First, in the *initial assignment* phase, we assign the zones to their target servers, i.e., find a target server for each client. Then, in the *refined assignment* phase, each client is assigned to an appropriate server to communicate with its target server, i.e., find a contact server for each client.

In fact, both the initial assignment and refined assignment are themselves complex optimization problems. In order to obtain good solutions to the CAP, we must optimize some “assignment costs” in both phases. In the following sections, we separately formulate the initial assignment and refined assignment problem, and prove them to be NP-hard.

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

4.3.1 The initial assignment problem (IAP)

To formulate the initial assignment problem, we propose the following metric to measure the cost of assigning a zone z_j to a server s_i

$$C_{ij}^I = |\{c_k \in z_j : d_{c_k s_i} > D\}| \quad (4.3.1)$$

C_{ij}^I measures the number of clients in a zone z_j that *do not* satisfy the delay bound D , i.e., without QoS. Therefore, by minimizing the total cost when all zones are assigned, the total number of clients with QoS in the DVE would be maximized. The initial assignment problem is formulated as follows.

Definition 4.3.1. Initial assignment problem (IAP)

Let $M = \{1, \dots, m\}$ and $N = \{1, \dots, n\}$ be the set of indexes of servers and zones in the system, respectively. For each $i \in M$ and $j \in N$, given the cost C_{ij}^I of assigning zone z_j to server s_i as defined in (4.3.1), find an assignment matrix $X = (x_{ij})$, with $x_{ij} = 1$ if zone z_j is assigned to server s_i or $x_{ij} = 0$ otherwise, which minimizes the total cost

$$C^I(X) = \sum_{i=1}^m \sum_{j=1}^n C_{ij}^I x_{ij} \quad (4.3.2)$$

subject to

$$\sum_{j=1}^n R_{z_j} x_{ij} \leq C_{s_i}, \forall i \in M, \quad (4.3.3)$$

$$\sum_{i=1}^m x_{ij} = 1, \forall j \in N, \quad (4.3.4)$$

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

$$x_{ij} \in \{0, 1\}, \forall i \in M, \forall j \in N. \quad (4.3.5)$$

Remark 4.3.1. In Definition 4.3.1, constraint (4.3.3) ensures that the capacity of each server will not be exceeded. Constraint (4.3.4) means that each zone is assigned to only one server.

Remark 4.3.2. In the IAP, the number of binary decision variables is mn , while the number of constraints is $m + n$, i.e., $O(n)$, since the number of zones n is usually much larger than the number of servers m .

Theorem 4.3.1. *The IAP is NP-hard.*

Proof. We consider in our model the special case where the resource capacity C_{s_i} of each server is the same. We further assume that the cost C_{ij}^I when assigning a zone $z_j, j \in N = \{1, \dots, n\}$ to server $s_i, i \in M = \{1, \dots, m\}$ has the special form of Δ^{i-1} , where $\Delta > 1$ is a constant, i.e. assigning any zone to server s_1 costs $C_{1j}^I = 1$, to server s_2 costs $C_{2j}^I = \Delta$, to server s_3 costs $C_{3j}^I = \Delta^2$, etc. Then the optimization goal of IAP in this special case would be to assign all the zones to the smallest possible number of P servers, with server indexes ranging from 1 to P . This is exactly the well-known Bin Packing problem [60], in which a bin corresponds to a server and an item corresponds to a zone. Since IAP generalizes the Bin Packing problem, which is NP-hard, IAP is also NP-hard.² □

²We would like to thank Gunther Raidl for helping with the NP-hardness of the Generalized Assignment Problem and hence this proof.

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

4.3.2 The refined assignment problem (RAP)

After the initial assignment phase, some clients may still be without QoS. The refined assignment phase will attempt to further increase the number of clients with QoS in the DVE. One possible approach is to exploit the well-provisioned inter-server connections, if they are available. We propose the following metric for the refined assignment to measure the cost of selecting server s_k as the contact server for a client c_j , where s_i is c_j 's target server

$$C_{ij}^R = \begin{cases} (d_{c_j s_k} + d_{s_k s_i}) - D, & \text{if } (d_{c_j s_k} + d_{s_k s_i}) > D; \\ 0, & \text{if } (d_{c_j s_k} + d_{s_k s_i}) \leq D. \end{cases} \quad (4.3.6)$$

C_{ij}^R measures the “distance” from the delay bound D of the communication delay of a client c_j if c_j is assigned to server s_k as its contact server. Therefore, by minimizing the total cost when all clients are assigned, the total number of clients with QoS in the DVE would be maximized. The refined assignment problem is then formulated as follows.

Definition 4.3.2. Refined assignment problem (RAP)

Let $M = \{1, \dots, m\}$ and $K = \{1, \dots, k\}$ be the set of indexes of servers and clients in the system, respectively. For each $i \in M$ and $j \in K$, given the cost C_{ij}^R of selecting server s_i as the contact server of client c_j , find an assignment matrix $X = (x_{ij})$, with $x_{ij} = 1$ if client c_j takes server s_i as its contact server or $x_{ij} = 0$ otherwise, which minimizes the total cost

$$C^R(X) = \sum_{i=1}^m \sum_{j=1}^k C_{ij}^R x_{ij} \quad (4.3.7)$$

subject to

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

$$\sum_{j=1}^k R_{c_j}^C x_{ij} \leq (C_{s_i} - \sum_{c_k} R_{c_k}^T), s_i = s^t(c_k), \forall i \in M, \quad (4.3.8)$$

$$\sum_{i=1}^m x_{ij} = 1, \forall j \in K, \quad (4.3.9)$$

$$x_{ij} \in \{0, 1\}, \forall i \in M, \forall j \in K. \quad (4.3.10)$$

Remark 4.3.3. In Definition 4.3.2, constraint (4.3.8) ensures that the *remaining* capacity of each server, i.e., the amount of resource of each server that still remains after the initial assignment, will not be exceeded. Constraint (4.3.9) means that each client is assigned to only one contact server.

Remark 4.3.4. In the RAP, the number of binary decision variables is mk , while the number of constraints is $m + k$, i.e., $O(k)$, since the number of clients k is usually much larger than the number of servers m .

Theorem 4.3.2. *The RAP is NP-hard.*

Proof. The proof is similar to that for the IAP. □

4.4 Related work

4.4.1 Client-to-Server assignment in DVEs

To the best of our knowledge, there is no existing work that directly addresses the client assignment problem in DVEs as we have described. Research on how to assign clients to servers in DVEs is usually formulated as a load distribution problem in a locally

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

distributed server architecture, i.e., all the servers are placed in the same machine room [38, 74, 80, 84, 104]. Such approaches may damage the interactivity of the DVE, since clients may be far away (in terms of network delays) from the servers.

Recently, the interest in investigating server and network architectures to reduce the effect of network latency for DVEs has been increased. In [86, 98], the authors introduced the concept of *latency driven distribution* (LDD) for the distribution of a DVE over the networking architecture, as opposed to the traditional *resource driven distribution* (RDD), i.e., load distribution. Our client assignment problem shares the idea of the LDD concept. However, the authors of [86, 98] only investigate the provision of immersive audio communication in DVEs within the context of the LDD concept. In their work, two clients that belong to two adjacent DVE zones are able to hear each other. However, several important kinds of interactions that may happen in a DVE such as shooting enemies, manipulating objects, changing object ownership, etc., were not studied. These interactions may raise complicated consistency issues, if the clients in adjacent zones are managed by different physical server machines.

In a more recent work [73], the authors proposed a distributed algorithm for clients to select the best server in a mirrored architecture for online games, taking into account the network delay between clients and servers. The mirrored architecture replicates the DVE zones at multiple servers. This approach shares some similarities with the web server replica placement problem in CDNs [44, 93]. However, unlike the replication of web documents, DVE replication faces serious consistency problems [111] which may damage the users' experience in interacting with the virtual world. In our approach, which employs the discrete partitioning strategy, only one server has the control over the

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

state of a zone, thus consistency can be enforced.

4.4.2 Classical combinatorial optimization problems

Bin Packing problem

We have proved that our two sub-problems IAP and RAP are both NP-hard by showing that our problems generalize the Bin Packing problem [6, 60]. The Bin Packing problem is a classical combinatorial NP-hard problem with many practical applications, such as filling up containers, loading trucks, etc. Informally stated, this problem consists of packing objects of different volumes into a finite number of bins. The capacities of the bins are all the same. The objective of the Bin Packing problem is then to minimize the total number of bins used.

Since this problem is NP-hard, efficient heuristic methods are often required instead of optimal solutions. Among the most well-known heuristics are the Best-Fit-Decreasing (BFD) and First-Fit-Decreasing (FFD) approaches [60]. Both of these approaches use no more than $\frac{11}{9}\delta + 4$ bins, where δ is the optimal number of bins.

Generalized Assignment problem

The objective of the Generalized Assignment Problem (GAP) [51, 96] is to minimize the total cost of the assignment of jobs to machines, where each job is assigned to exactly one machine subject to the capacity constraints of the machines. The GAP was motivated by many practical problems, such as assigning jobs to computers in a network of workstations, etc. The problem can be formulated as an Integer Linear Programming (ILP) problem as follows:

Definition 4.4.1. Generalized assignment problem (GAP)

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

Let $I = \{1, \dots, m\}$ and $J = \{1, \dots, n\}$ be the set of machines and jobs, respectively. For each machine i we are given a resource capacity b_i . For each $i \in I$ and $j \in J$ we are given a cost c_{ij} and a resource requirement a_{ij} for assigning job j to machine i . The objective is to find an assignment matrix $x = (x_{ij})$, with $x_{ij} = 1$ if job j is assigned to machine i or $x_{ij} = 0$ otherwise, which minimizes the total cost

$$C(x) = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (4.4.1)$$

subject to

$$\sum_{j=1}^n a_{ij} x_{ij} \leq b_i, \forall i \in I, \quad (4.4.2)$$

$$\sum_{i=1}^m x_{ij} = 1, \forall j \in J, \quad (4.4.3)$$

$$x_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J. \quad (4.4.4)$$

From the above formulation, we can see that our problems IAP and RAP are very similar to the GAP. The main difference between our problem and the GAP lies in the resource requirement definition. In our problems, for example the IAP, the resource requirement of a zone is constant regardless of which server the zone is assigned to. On the contrary, the resource requirement in the GAP is varied for each pair of job and machine, i.e., a job j will have different resource requirements of a_{ij} or a'_{kj} if it is assigned to machine i or k , respectively.

The GAP is NP-Hard in the strong sense [96], since the decision problem associated

CHAPTER 4. CLIENT ASSIGNMENT PROBLEM

with the feasibility of the GAP is a NP-complete problem. Several efficient heuristics have been developed for the GAP [51, 96].

4.5 Summary

In this chapter, we have formulated the client assignment problem (CAP). We then present our two-phase approach to the CAP, which consists of an initial assignment phase and a refined assignment phase. Both phases are then proved to be NP-hard using a reduction to the well-known Bin Packing problem.

In the next chapter, we shall adapt a well-known heuristics for the GAP to help solving our client assignment problem.

Chapter 5

Client assignment algorithms

To address the client assignment problem (CAP), in this chapter we propose some algorithms for the initial assignment (IAP) and then for the refined assignment (RAP). Since both IAP and RAP are NP-hard, we seek heuristic solutions instead of optimal ones. We start with algorithms for the IAP, with which we assign zones to servers, i.e., determine target servers for clients. We propose three algorithms for the IAP: the first one randomly assigns the zones, while the second one is a greedy heuristics to minimize the number of clients without QoS in the system. The third algorithm for IAP differs from the second one only in the cost metric used.

After the target servers for clients are determined, in the refined assignment phase, we find an appropriate contact server for each client. This assignment phase depends on the quality, i.e., network latency, of the inter-server connections. If the inter-server connections are not well-provisioned, i.e., the servers use the commodity Internet to communicate with each other, the only feasible refined assignment approach is to connect each client to the server hosting its zone, i.e., based on the virtual location of clients. On the other hand, if servers are fully meshed with good network connections, e.g., private,

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

dedicated networks, or QoS-enable networks, then it is possible to reduce the client-server communication delay of a client by assigning that client to a contact server different from its target server. In this case, we propose two algorithms. The first algorithm selects the closest server of a client as its contact server, and the second algorithm is a greedy heuristics similar to the one for solving the IAP.

Finally, by combining the algorithms for the IAP and the RAP, we have the two-phase algorithms for the CAP. For comparison purpose, we also present the optimal algorithm using the “branch-and-bound” method to solve the integer programming formulation of IAP and RAP.

5.1 Algorithms for the IAP

5.1.1 Random assignment of zones (RanZ)

In RanZ, zones are assigned to randomly selected servers with the only concern of not overloading the servers. The pseudo-code of the algorithm is shown in Fig. 5.1. In this algorithm, the following procedure is repeated until all zones have been assigned: first we randomly select a zone z_j , and then a server s_i with sufficient capacity is randomly selected to take z_j , i.e., the target server of all clients in z_j is set to s_i .

Remark 5.1.1. The computational cost of the RanZ algorithm is $O(mn)$, where n is the number of zones and m is the number of servers.

5.1.2 Greedy assignment of zones - algorithm 1 (GreZ-1)

Since RanZ is oblivious to client-server network delays when assigning zones to servers, the obtained performance in terms of the number of clients with QoS may not be good,

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

```

1 begin
2    $R_{s_i} = 0$ ;
3   while  $Z \neq \emptyset$  do
4     randomly select a zone  $z_j$  from  $Z$ ;
5     randomly select a server  $s_i$ , with  $R_{s_i} + R_{z_j} \leq C_{s_i}$ ;
6     assign  $z_j$  to  $s_i$ ;
7      $R_{s_i} = R_{s_i} + R_{z_j}$ ;
8     remove  $z_j$  from  $Z$ ;
9   end
10 end

```

Figure 5.1: IAP - Random assignment of zones

```

1 begin
2   foreach  $c_k \in C$  do
3     foreach  $s_i \in S$  do
4       if  $d_{c_k s_i} > D$  then
5          $C_{ij}^I = C_{ij}^I + 1$ ,  $c_k \in z_j$ ;
6       end
8     end
9   foreach  $z_j \in Z$  do
10    sort the list  $L_j^M$  of values  $\mu_{ij} = -C_{ij}^I$  in descending order,  $s_i \in S$ ;
11    find  $\rho_j = \mu_{i_j j} - \max_{s \neq i_j} \mu_{sj}$ , where  $i_j = \arg \max_i \mu_{ij}$ ;
12  end
13  sort the list of zones  $Z$  according to the value  $\rho_j$  of each zone  $z_j$  in descending order;
14  while  $Z \neq \emptyset$  do
15    pick the first zone  $z_j \in Z$ ;
16    while  $L_j^M \neq \emptyset$  do
17      pick the first desirability  $\mu_{ij} \in L_j^M$ ;
18      pick server  $s_i$ ;
19      if  $R_{s_i} + R_{z_j} \leq C_{s_i}$  then
20        assign  $z_j$  to  $s_i$ ;
21         $R_{s_i} = R_{s_i} + R_{z_j}$ ;
22        remove  $z_j$  from  $Z$ ;
23        break;
24      end
25      remove  $\mu_{ij}$  from  $L_j^M$ ;
26    end
27  end
28 end

```

Figure 5.2: IAP - Greedy assignment of zones - algorithm 1

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

i.e., the cost of the assignment as defined in Equation (4.3.2) may be high. Hence, in the GreZ-1 algorithm, we use a greedy heuristics to minimize the total number of clients *without* QoS in the system.

The pseudo-code is shown in Fig. 5.2. Let $\mu_{ij} = -C_{ij}^I$ (recall that C_{ij}^I measures the number of clients without QoS in zone z_j if z_j is assigned to s_i) be a heuristic measure of the *desirability* of assigning zone z_j to server s_i . The smaller the cost C_{ij}^I is, the higher the desirability μ_{ij} is. The algorithm iteratively considers all the unassigned zones and pick a zone z_j with the maximum difference ρ_j between the largest desirability μ_{ij} and the second largest desirability μ_{sj} . Then, z_j is assigned to a server s_i with the highest value of μ_{ij} and with sufficient resource capacity. This procedure is adapted from the well-known approach used to solve the Generalized Assignment Problem [51, 96].

Let us illustrate the effectiveness of the GreZ-1 algorithm using an example. Assuming that the DVE has two zones z_1 and z_2 , and two servers s_1 and s_2 . Each server can only take one zone, due to its capacity constraint. Let us further assume that assigning z_1 to s_1 or s_2 costs $C_{11}^I = 10$ or $C_{21}^I = 20$, respectively. Similarly, assigning z_2 to s_1 or s_2 costs $C_{12}^I = 5$ or $C_{22}^I = 10$, respectively.

The desirability difference ρ_1 of zone z_1 is equal to $\mu_{11} - \mu_{21} = -C_{11}^I - (-C_{21}^I) = -10 - (-20) = 10$, while ρ_2 of z_2 is equal to $\mu_{12} - \mu_{22} = -C_{12}^I - (-C_{22}^I) = -5 - (-10) = 5$. Hence, the GreZ-1 algorithm will select z_1 to assign first. As a result, z_1 is assigned to server s_1 , and z_2 is assigned to server s_2 , since each server can only take one zone. The total cost of this assignment is $10 + 10 = 20$, i.e., in total there are 20 clients that are without QoS in the system.

However, if we do not follow the procedure of GreZ-1, then z_2 may be chosen to assign

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

first. In this case, z_2 will be assigned to s_1 , thus z_1 has to be assigned to s_2 . The total cost in this case would be $20 + 5 = 25$, which is larger than the cost obtained using GreZ-1. Thus, by sorting the zones according to their desirability differences ρ , GreZ-1 can effectively reduce the number of clients without QoS in the system.

Complexity analysis of GreZ-1

We analyze the computational cost of the GreZ-1 algorithm. Let m , n and k denote the number of servers, the number of zones and the total number of clients, respectively. The GreZ-1 algorithm consists of four parts. The first part (from line 2 to line 8) is to find all the values of μ_{ij} . The cost of this part is $O(mk)$. The second part (from line 9 to line 12) is to find ρ_j for all $z_j \in Z$. In the line 10, Quicksort is used. Hence, the cost of this part is $O(n)[O(m \log m) + O(1)]$, or $O(mn \log m)$.

The third part (line 13) is to sort the list of n values of ρ_j . Using Quicksort, the cost is $O(n \log n)$. The final part (from line 14 to line 27) is to assign zones to servers. There are m servers and n zones, hence the cost of this part is $O(nm)$.

Hence, the overall computational cost of the GreZ-1 algorithm is $O(mk) + O(mn \log m) + O(n \log n) + O(nm)$, or $O(\max\{O(mk), O(mn \log m), O(n \log n)\})$.

Remark 5.1.2. The complexity of the GreZ-1 algorithm is $O(\max\{O(mk), O(mn \log m), O(n \log n)\})$, where n is the number of zones, m is the number of servers and k is the number of clients in the system. In practice, usually $k \gg m$ and $k \gg n$, hence the overall complexity of GreZ-1 for most cases can be written as $O(mk)$.

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

5.1.3 Greedy assignment of zones - algorithm 2 (GreZ-2)

The only difference between GreZ-1 and GreZ-2 lies in the cost metric they use. With the cost metric in Equation (4.3.1), Greedy-1 aims to minimize the total number of clients without QoS in the system directly. For the GreZ-2 algorithm, we propose to use the following metric to measure the cost when assigning a zone z_j to a server s_i

$$C'_{ij} = \frac{\sum_{c_k \in z_j} d_{c_k s_i}}{|z_j|} \quad (5.1.1)$$

C'_{ij} measures the average delay from all client $c_k \in z_j$ to server s_i if they are all assigned to s_i . By minimizing this cost function, it is expected that the selected server s_i for z_j will be near (in terms of network delay) to the center of mass of the client population of z_j , thus the number of clients with QoS may become large.

5.2 Algorithms for the RAP

5.2.1 “Normal” inter-server networks

In this case, we assume that the inter-server networks are not well-provisioned, i.e., the communication between any two servers in the system is done via the commodity Internet (hence the name “normal” inter-server networks). Hence, there’s no incentive to forward messages from a client to its target server via another contact server. We propose an algorithm, referred to as the *Virtual Location based assignment of clients*, for the RAP.

Virtual Location based assignment of clients (VirC)

The VirC algorithm for the RAP adopts the most “natural” way to assign clients to servers in DVEs. It only considers the virtual location of each client c_j when determining

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

a contact server for c_j , thus c_j will connect to the same server that is also hosting c_j 's zone, i.e., the contact server of c_j is the same as its target server. This approach will not incur any inter-server communication cost. However, the number of clients with QoS is not improved compared to the initial assignment.

Remark 5.2.1. The complexity of the VirC algorithm is $O(k)$, where k is the number of clients in the system.

5.2.2 Well-provisioned inter-server networks

In this case, we assume that the inter-server networks are well-provisioned with low network delays. We propose two algorithms for the RAP, namely the *Closest-server assignment of clients*, and the *Greedy assignment of clients*.

Closest-server assignment of clients (CloC)

In the CloC algorithm, the closest server of each client is chosen as its contact server. The main motivation of this algorithm is to reduce the client-target server communication delay by exploiting the well-provisioned inter-server networks which have low delays.

Remark 5.2.2. The complexity of the CloC algorithm is $O(km)$, where k is the number of clients and m is the number of servers in the system.

Greedy assignment of clients (GreC)

The GreC algorithm is a greedy heuristics which takes into account network delay from a client to its target server when selecting contact server for the client. The pseudo-code is shown in Fig. 5.3. This algorithm is similar to the GreZ-1 or GreZ-2 algorithm for the

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

IAP. The major difference between the two algorithms lies in the cost metric used. GreC uses the cost metric defined in Equation (4.3.6).

```

1 begin
2   foreach  $c_j \in C$  do
3     get target server  $s_i$  of  $c_j$ ;
4     if  $d_{c_j s_i} \leq D$  then
5       select  $s_i$  as the contact server of  $c_j$ ;
6     else
7       add  $c_j$  to the list  $L^E$ ;
8     end
9   end
10  foreach  $c_j \in L^E$  do
11    find  $\mu_{ij} = -C_{ij}^R, \forall s_i \in S$ ;
12    sort the list  $L_j^M$  of values  $\mu_{ij}$  descendingly;
13    find  $\rho_j = \mu_{ij} - \max_{s \neq i_j} \mu_{sj}$ , where  $i_j = \arg \max_i \mu_{ij}$ ;
14  end
15  sort the list  $L^R$  of values  $\rho_j$  in descending order;
16  while  $L^E \neq \emptyset$  do
17    pick client  $c_j$  with highest  $\rho_j$ ;
18    while  $L_j^M \neq \emptyset$  do
19      pick the highest desirability  $\mu_{ij} \in L_j^M$ ;
20      pick server  $s_i$ ;
21      if  $R_{s_i} + R_{c_j}^C \leq C_{s_i}$  then
22        select  $s_i$  as the contact server of  $c_j$ ;
23         $R_{s_i} = R_{s_i} + R_{c_j}^C$ ;
24        remove  $c_j$  from  $L^E$ ;
25        break;
26      end
27      remove  $\mu_{ij}$  from  $L_j^M$ ;
28    end
29  end
30 end

```

Figure 5.3: RAP - Greedy assignment of clients

GreC starts by considering the round-trip delay to target server of each client in the system. If the network delay from a client c_j to its target server s_i is less than the delay bound, the algorithm selects the same server s_i as the contact server of c_j . Otherwise, c_j is added to a list L^E , which only contains clients having network delay larger than the

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

delay bound. The next part of the algorithm attempts to assign each client in L^E to a contact server to increase the number of clients with QoS in the system. This part is similar to the GreZ algorithm, except for the cost function used.

Complexity analysis of GreC

We analyze the computational cost of GreC algorithm. Let m, k denote the number of servers and the total number of clients, respectively. The GreC algorithm consists of four parts. The first part (line 2 to line 9) is to construct a list of clients that are without QoS and assign contact servers for clients with QoS. The cost of this part is $O(k)$. The second part (line 10 to line 14) is to construct the list of values ρ_j for all clients that are without QoS. The cost of this part is $O(k)[O(m) + O(m \log m) + O(1)]$ or $O(km \log m)$. The third part (line 15) costs $O(k \log k)$ (using Quicksort). The final part (line 16 to 29) is to assign clients to contact servers. The cost of this part is $O(km)$.

Hence the overall computational cost of the GreC algorithm is $O(k) + O(km \log m) + O(k \log k) + O(km)$ or $O(\max\{m \log m, \log k\}k)$.

Remark 5.2.3. The complexity of the GreC algorithm is $O(\max\{m \log m, \log k\}k)$, where m is the number of servers and k is the number of clients in the system.

5.3 Two-phase algorithms for the CAP

A two-phase algorithm for the CAP is obtained by combining the algorithms for the IAP and the RAP. If the inter-server networks are not well-provisioned, we have the following three two-phase algorithms:

- RanZ/VirC

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

- GreZ-1/VirC
- GreZ-2/VirC

If the inter-server networks are well-provisioned, we have the following six different two-phase algorithms:

- RanZ/CloC
- RanZ/GreC
- GreZ-1/CloC
- GreZ-1/GreC
- GreZ-2/CloC
- GreZ-2/GreC

5.4 Optimal algorithm using branch-and-bound

For comparison purposes, based on the Integer Programming formulation of the IAP and RAP, we use the so-called “branch-and-bound” method [7] implemented in the free Mixed Integer Linear Programming (MILP) solver `lp_solve` [14] to obtain the optimal solutions to the CAP. `lp_solve` is a free MILP solver with full source code, examples and detailed manuals. It can solve pure linear, mixed integer/binary, semi-continuous and special ordered sets models. Although it is written in ANSI C, it can be called as a library from various languages like C, VB, .NET, Delphi, Excel, Java (through JNI interface).

We should note that this approach is only applicable when the system size is small, otherwise the running time of `lp_solve` will become very long (on the order of several hours), which is clearly impractical for highly interactive applications like DVEs.

5.5 Practical considerations

In this section we address some practical considerations in implementing the proposed assignment algorithms. These considerations include how to obtain input data for the algorithms and how to deal with the inherent dynamics of DVEs.

5.5.1 Obtaining input data

The first issue is how to obtain input data for the proposed assignment algorithms. Our algorithms are centralized, hence we need a dedicated server¹ to collect the data and run the algorithms. The input data includes the client-server and inter-server round-trip network delays, and the resource requirement of each client on its server.

Network delays

The server that runs our algorithms can obtain the network delays using some well-known network measurement methods such as IDMaps [53], King [56] and GNP [85]. These methods are scalable and incur little measurement overhead.

- **Internet Distance Map Service (IDMaps):** IDMaps is a scalable Internet-wide architecture for measuring and disseminating distance information on the Internet. It was designed to be the underlying service that provides the necessary information used by SONAR/HOPS [81], which is a simple protocol for a host to quickly and efficiently learn the network distance between any two hosts. IDMaps relies on special end hosts called *tracers* deployed at some strategic locations in the Internet. Each tracer measures the distance between itself and a set of Internet hosts that are close to it. The measured

¹This server can be the one that the clients first connect to before joining the virtual world. It is regarded as the master server, which monitors the status and locations of other servers that are hosting the zones, and manages users' related information such as username, password, etc.

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

distances are then sent to a set of dedicated servers, called SONAR/HOPS servers.

To estimate the distance between any two Internet hosts, the SONAR/HOPS servers calculate the sum of the distances from each host to its nearest tracer, and the distance between the two corresponding tracers. Obviously, the accuracy of such approach depends on the number and Internet locations of tracers. The goal of IDMaps is to achieve an estimation accuracy “within a factor of 2 with very high probability and often better than that” [53]. Any client can then query a SONAR/HOPS server to learn the distance between any two other hosts using client-server query/reply protocols similar to the widely used DNS query/reply protocol.

- **Global Network Positioning (GNP):** GNP [85] estimates network latencies among Internet hosts using absolute “coordinates” computed from modelling the Internet as a M -dimensional space. To enable the scalable computation of host coordinates in the Internet, a small set of dedicated hosts called “landmarks” first select their own coordinates in the chosen geometric space. These landmarks’ coordinates are then distributed to any host that wants to compute its own coordinates.

A host can compute its coordinates by first measuring the distance between itself and the landmarks, then using the landmarks’ coordinates to derive its own coordinates. Based on the coordinates of any two hosts, we can estimate the latency between them as the Euclidean distance between their coordinates. This approach is scalable, since estimating pair-wise network distances between N hosts in GNP only requires NK measurements, where $K \ll N$ is the number of landmarks, compared to the straightforward approach (let the hosts measure the network distances to each other) which require $\frac{1}{2}N^2$

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

measurements [103]. The authors of GNP claimed that their approach is more accurate than IDMaps. More specifically, in 90% of their experimental cases, the latency estimations produced by GNP are within the range of $[\frac{2}{3}d, \frac{3}{2}d]$, where d is the “true” latency.

We should note that GNP and IDMaps are similar to each other in the sense that they both requires the deployment of additional infrastructures, e.g., tracers and landmarks. In addition, they do not directly measure the latencies between the hosts, rather, they “calculate” the latencies from a small number direct measurements.

- **King:** King [56] is a very clever approach to scalable network latency measurements. It does not require any additional infrastructure like IDMaps or GNP. Instead, King uses existing recursive DNS queries to accurately measure round-trip network delays between arbitrary Internet end hosts.

Given a pair of hosts, King first finds their two nearby DNS servers, for example a and b . It is noted that Internet hosts are usually close their respective DNS servers. By issuing a recursive DNS query to one of these name servers, for example the name server a , King is able to measure a latency value which is the latency between a and b plus the latency between the machine that is running it², says c , and the name server a . Then, it measures the latency between c , and the name server a using an ICMP ping. After subtracting the first latency value by the second one, King can produce a rather accurate estimation of latency between the two given hosts.

We should note that since King uses direct measurements, its accuracy may be significantly better than extrapolation-based approaches like IDMaps and GNP. In fact, in

²This machine is also the server that executes our assignment algorithms.

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

their evaluation, the authors of King claimed that their tool was able to produce latency estimations with less than 20% error in nearly 80% of all cases.

Resource requirement

In this study, the server resource requirement of each client is measured as the bandwidth requirement. Following the work in [92], the bandwidth requirement of each client on its target server $R_{c_i}^T$ in a server-based architecture can be estimated in advance as follows.

First we assume that the size of the message each client c_i sends to its target server is L bytes, the client's sending frequency, i.e., frame rate, is T , and the number of concurrent clients in c_i 's zone is N . On receiving a message from client c_i , the target server replies c_i with the state updates of *all* the clients in c_i 's zone, which requires LN bytes.³ Hence, the bandwidth requirement of a client c_i on its target server $R_{c_i}^T$ can be calculated as

$$R_{c_i}^T = (L + LN)T = (N + 1)LT \quad (5.5.1)$$

Based on the definition in the previous chapter, the bandwidth requirement of a client c_i on its contact server can be calculated as

$$R_{c_i}^C = \begin{cases} 2(N + 1)LT, & \text{if } s^c(c_i) \neq s^t(c_i); \\ 0, & \text{if } s^c(c_i) = s^t(c_i). \end{cases} \quad (5.5.2)$$

The total bandwidth requirement on a server s_j can then be easily calculated from the number of clients that are interacting on it (i.e., it is the target server for these clients), and the number of clients that are connecting to it (i.e., it is the contact server for these clients).

³For simplicity, we assume that all messages have the same size. This assumption is also used in [92].

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

5.5.2 Dynamic executions of assignment algorithms

Another important consideration is the dynamic property of DVEs. During the course of interactions in the virtual world, clients may move from one zone to another, new clients may join, existing clients may also leave the virtual world. An obtained client assignment may not be good after some time. Thus, the proposed two-phase algorithm needs to be executed from time to time to ensure good client assignments.

We should note that dynamic execution of the client assignment algorithms may incur inter-server bandwidth consumption due to the migrations of zones or clients across servers. Nevertheless, we believe this cost is much smaller compared to the bandwidth consumption due to the interactions of users in the DVE. This suggests that the bandwidth cost of re-executing the assignment algorithms is not a big concern. Indeed, a similar assumption is used in the context of the Web replica placement problem [93].

Moreover, in a recent study [66] of the very popular MMOG Lineage II developed by NCsoft, Korea, the authors found that the average value of session durations, i.e., the average time a player keeps on playing the game, is about 3 hours, which is rather long. This may indicate that we do not have to execute the assignment algorithms frequently.

5.6 Client assignment versus server placement

Up to now, we have proposed several server placement approaches and client assignment algorithms. They all serve the same purpose: enhancing the interactivity for DVEs. Apparently, each method has its own pros and cons. In this section, we discuss some relations between these two methods.

- Server placement and client assignment can *complement* to each other in enhancing

CHAPTER 5. CLIENT ASSIGNMENT ALGORITHMS

the interactivity of DVEs, which is the ultimate goal of our research. For example, if all the servers are already placed at some good network locations, then appropriate client assignment algorithms can be used to further enhance the interactivity of the DVE. On the other hand, a good server placement certainly helps to maximize the performance of client assignment algorithms.

- We believe that client assignment algorithms are more *flexible* than server placement approaches in the sense that the former does not require any special infrastructure to be deployed. As we have already discussed in previous chapters, placing servers at good network locations is not always achievable, due to several technical and economic constraints. In such scenarios, efficient client assignments are highly desirable to enhance the interactivity of DVEs. As we will show later in this thesis, good client assignments can significantly lift up the interactivity level, even if the servers are placed at some bad network locations, i.e., the routers with the lowest node degrees. Hence, to some extent, we may say that client assignments are of greater importance than server placements, although it would be highly advantageous to combine the effectiveness of both methods.

5.7 Summary

In this chapter, we have proposed and analyzed several algorithms for the client assignment problem. We have also addressed a number of practical considerations, including how to obtain network latencies, server resource requirements and dynamic executions, in implementing the proposed algorithms.

In the next chapter, we shall describe in detail our methodology in evaluating server placement approaches and client assignment algorithms.

Chapter 6

Evaluation methodology

In this chapter, we present our methodology in evaluating the effectiveness of various server placement approaches and client assignment algorithms. In the evaluation, we strive to be as realistic as possible, by using a wide range of network topologies, including both real and synthetic ones. We model the network latencies based on existing real Internet measurements and studies [89, 106]. In addition, for the workloads, based on recent studies of real online game systems, e.g., [52], we simulate various client distributions (clustered, uniform, etc.) both in the physical world (the network) and the virtual world.

6.1 Network models

6.1.1 Network topologies

In our simulations, we use both synthetic topologies generated by the popular Internet topology generator BRITE [8], and real Internet topologies. BRITE is a widely used

CHAPTER 6. EVALUATION METHODOLOGY

Figure 6.1: BRITE topology visualized by Otter

topology generator from Boston University, which is regarded as a *universal topology generator* with the design principles including representativeness, inclusiveness, and interoperability. Representativeness implies that the generated topologies can accurately reflect many aspects of the actual Internet topology, for example hierarchical structure, degree distribution, etc. Inclusiveness means that it is able to combine many topology generation models in a single generation tool, thus reducing learning efforts for users. BRITE implementation provides several popular generation models, e.g., Waxman, power-law, hierarchical, etc. Finally, interoperability means that the generated topologies can be used in a wide range of network simulation tools, such as ns-2 [24], SSF [22] and OM-NeT++ [17], as well as visualization applications, such as Otter [1] (see Fig. 6.1).

To strengthen our simulation results, we also collect some real Internet topologies from the SSF web site [22], and the NLANR web site [25], which itself collected its data

CHAPTER 6. EVALUATION METHODOLOGY

from the University of Oregon Route Views project [20]. While these topologies are not complete, they at least partially reflect the “true” topology of the Internet, which may have great impacts on the simulation results.

The topologies used in our simulations include both flat topology models and hierarchical topology models as follows.

Flat topologies

- **Waxman topology:** In this topology generation model, the nodes in the network are placed at random positions in a two-dimensional plane. Then, the Waxman’s probability model [108] is used to interconnect any two nodes u and v in the network as follows:

$$P(u, v) = \alpha e^{-d/(\beta L)} \quad (6.1.1)$$

where $0 < \alpha, \beta \leq 1$ are the parameters of the model, d is the Euclidean distance from node u to node v , and L is the maximum distance between any two nodes. Put another way, the Waxman model considers all pairs of nodes, and then decides whether to add a link between any two nodes with a probability that depends on the distance between these two nodes and the longest distance between any two nodes in the network.

In the simulations, we use BRITE to generate a Waxman topology with 3000 nodes and 6000 links.¹

- **Power-law topology:** In the seminal paper by Faloutsos et al. [50] in 1999, the authors used real network measurements of the router-level and AS-level Internet topologies to show that these topologies exhibit power-laws of the form $y \propto x^\alpha$ for the outdegree of

¹Note that all the Waxman topologies generated in the simulation use the default parameters given by BRITE: $\alpha = 0.15$, $\beta = 0.2$.

CHAPTER 6. EVALUATION METHODOLOGY

a node versus its rank,² and the frequency of an outdegree³ versus that outdegree. Let d_v , r_v and f_d denote the outdegree of a node v , the rank of node v and the frequency of the outdegree d , respectively. The power-law for the outdegree of a node is expressed as follows:

$$d_v \propto r_v^R \quad (6.1.2)$$

where R is a constant, called the *rank exponent*. This power-law states that the outdegree of a node is proportional to the rank of the node to the power of a constant R .

The power-law for outdegree frequency is expressed as follows:

$$f_d \propto d^\Theta \quad (6.1.3)$$

where Θ is a constant, called the *outdegree exponent*. This power-law simply states that the frequency of an outdegree d is proportional to d to the power of a constant Θ .

The measurement-based study in [50], which used several snapshots of the Internet at different times and obtained seemingly invariant results, suggests that these power-laws are fundamental properties of Internet topologies. However, existing network topology generation models, such as Waxman, do not capture these properties. In addition, the true reasons behind these properties have been the subject of much research. In 1999, Barabasi and Albert [33] proposed a model that can suggest some explanations for the power-law in the frequency of outdegree in network topologies. There are two possible reasons: *incremental growth* and *preferential connectivity*. More specifically, incremental growth refers to networks that expand continuously by the addition of new nodes, and

²The rank of a node in a network topology is its index in the order of decreasing outdegree.

³The frequency of an outdegree is the number of nodes with this outdegree.

CHAPTER 6. EVALUATION METHODOLOGY

preferential connectivity refers to the tendency of new nodes to attach preferentially to existing nodes that are already well connected.

In the implementation of the Barabasi-Albert model in BRITE, the nodes are interconnected according to the incremental growth approach. When a node u joins the network, the probability that it connects to a node v already belonging to the network is given by:

$$P(u, v) = \frac{d_v}{\sum_{k \in V} d_k} \quad (6.1.4)$$

where d_v is the degree of node v , V is the set of nodes that have joined the network and $\sum_{k \in V} d_k$ is the total of the outdegrees of all nodes that have already joined the network.

For the simulations, we use BRITE to generate a power-law topology using the Barabasi-Albert model with 3000 nodes and 5997 links.

- **Real topology:** To complement the synthetic topologies generated by BRITE, we also use a real Internet topology collected from NLNR [25]. This topology has 3024 nodes and 5192 links.

Hierarchical topologies

Today's Internet exhibits hierarchical structure in the sense that it can be viewed as a collection of multiple Autonomous Systems (ASes) interconnected via complex peering agreements.⁴ Each AS consists of multiple nodes (routers, switches, and hosts) interconnected with each other. Hence, synthetic topologies should be generated to reflect the hierarchical nature of the real Internet topologies. Flat models such as Waxman and Barabasi-Albert do not directly aim at generating hierarchical topologies, hence, other

⁴Peering is the practice of exchanging Internet traffic between the networks of different ISPs [27].

CHAPTER 6. EVALUATION METHODOLOGY

generation models may be required.

In our simulations, we use BRITE’s top-down approach to generate hierarchical topologies. The generated topologies are two-level, i.e., first BRITE generates an AS-level topology using a flat generation model such as Waxman or Barabasi-Albert. Then, for each node in the AS-level topology, BRITE generates a router-level topology using one of the available flat generation models. To interconnect the generated router-level topologies, BRITE utilizes the connectivity information among ASes.

- **Synthetic, hierarchical topology:** We generate a two-level topology that consists of 3000 nodes and 6197 links using BRITE. The topology has 100 ASes using Barabasi-Albert model, and each AS has 30 nodes using Waxman model.
- **Real, hierarchical topology:** For diversity, we collect a real AS-level topology (generated by processing BGP⁵ routing tables) with 110 nodes from [22], and use the DFN (German Research Network) topology [57] with 30 nodes as the router-level topology to construct a realistic hierarchical topology with 3300 nodes and 13442 links.

Table 6.1 summarizes the network topologies that are used in our simulations.

Table 6.1: Network topologies used in the simulations

Topology	Nodes	Links
Flat, Waxman (F-W)	3000	6000
Flat, Barabasi-Albert (F-BA)	3000	5997
Flat, real (F-R)	3024	5192
Hierarchical, Barabasi-Albert/Waxman (H-BA/W)	3000	6197
Hierarchical, real/real (H-R/R)	3300	13442

⁵BGP stands for Border Gateway Protocol, which is the core routing protocol of the Internet.

CHAPTER 6. EVALUATION METHODOLOGY

6.1.2 Network delay model

For simplicity, we assume that the round-trip network delay between any two nodes in the network topology is proportional to the number of link-hops between them. This assumption is similar to the one used in most of the previous work [90, 93]. In fact, a recent Internet measurement [89] also showed that round-trip delay is well-correlated with network hop counts.

To obtain more realistic simulations results, in calculating the network delay, we use both shortest-path routing and AS-level hierarchical routing [106] where possible. More specifically, when the network topology is a hierarchical one, AS-level hierarchical routing can be employed. However, when flat topologies are used, shortest-path routing is the only choice.

6.1.3 Shortest-path routing

The shortest-path routing strategy can be applied to both flat and hierarchical topologies, however in this study we use it only for flat topologies, e.g., Waxman, Barabasi-Albert, etc. The well-known Dijkstra algorithm is used to calculate the shortest distances (in terms of network delays) from a node to any other nodes in the given topology.

6.1.4 AS-level hierarchical routing

The AS-level hierarchical routing is a more realistic routing strategy for our simulations, since to some extent it may reflect the “true” routing practice in the current Internet [106]. With this strategy, first we calculate the AS-level shortest path. Then, to get the distance between any two nodes at the router-level, the AS-level shortest path is followed,

CHAPTER 6. EVALUATION METHODOLOGY

and for each AS on that shortest path, the router-level shortest path (within that AS) is used.

6.2 Workload models

6.2.1 Client distributions

Table 6.2: Distribution types

Type	Clusters in PW	Cluters in VW
0	No	No
1	Yes	No
2	No	Yes
3	Yes	Yes

To fully evaluate the performance of the proposed assignment algorithms and placement approaches, we simulate several different client distributions in both the virtual world and the physical world (the network). The number of clients may be larger in some specific zones of the virtual world than others, due to the clustering of clients in some “hot” zones. For example, in online games, clients may be clustered in the zones with large amounts of game resources such as energy, gold, etc. In the physical world, due to the differences in time zones of geographically distributed clients, at a specific time, the number of online clients in the DVE may be quite different for different geographic regions [52].

We simulate the clustering behavior of clients in the virtual world by randomly selecting some zones to have more clients than other zones. To simulate the clustering of clients in the physical world, some nodes in the network topology are randomly selected

CHAPTER 6. EVALUATION METHODOLOGY

to have a larger number of clients than the rest nodes. We have simulated a large number of different scenarios by changing the number of clusters and number of clients in each clusters, and obtained similar results. Hence, in this study, we use the following method to generate different client distributions. For the clustered distribution in the virtual world, the number of clients in a clustered zone is 10 times larger than that in a non-clustered zone. The clustered distribution for the physical world is generated in a similar manner. Table 6.2 shows the combination of different virtual world (VW) and physical world (PW) distributions.

6.2.2 Physical world-virtual world correlation

In general, we should note that clients gathering in the same zone of a DVE may not necessarily close to each other in terms of their physical locations. On the other hand, it is natural to observe that clients that are close to each other in their physical locations (e.g., from the same country or the same geographic region) tend to gather in a specific zone of the virtual world due to their common cultural preferences. These phenomena may have great impacts on the performance of the proposed algorithms.

To model the correlation between clients' locations in the physical world and those in the virtual world, we use a correlation parameter δ , where $0 \leq \delta \leq 1$ [86]. The higher the value of δ is, the stronger the tendency for clients from the close geographic locations to gather in specific zones of the virtual world. In our simulations, the following method, which was suggested in [86], is used to generate the correlation between the clients' physical locations and virtual locations, based on the value δ :

- Let's assume that we have a correlation value $\delta = 0.7$. This value can be obtained

CHAPTER 6. EVALUATION METHODOLOGY

from real measurements of existing massively online gaming systems, like Lineage or Everquest. We also have the total number of clients and all client-to-zone relations, i.e., the client distribution in the virtual world, as well as the number of clients in each network node of the topology, i.e., the client distribution in the physical world.

- We choose in advance a number of network nodes, each node is “associated” to a zone. We assume that if the correlation value is very high, e.g., $\delta \approx 1$, almost all of the clients in a zone would be from the network node that is “associated” to that zone. For example, we may assume that, in a DVE environment with very high correlation value, most clients in Singapore would usually gather in a specific zone of the virtual world.

- For each client, we generate a random number r , $0 \leq r \leq 1$. If $r \geq \delta$, the client is from a randomly chosen network node. If $r < \delta$, the client is from the network node that is “associated” to the client’s zone. After we select a client from a network node, the number of clients in that node is subtracted by 1. In the case that we need to select a client from a given network node, but there’s no more client in that node, the clients in the closest network node will be considered and so on. In this way, the higher the correlation parameter is, the more clients in close geographic regions would group together in the same zone of the virtual world.

6.2.3 Bandwidth usage

To estimate bandwidth requirement as in [92], the message sending frequency of each client (i.e., frame rate) is set to 25 messages per second, and the size of each message is set to 100 bytes. These values closely reflect some real settings [29].

CHAPTER 6. EVALUATION METHODOLOGY

6.3 Performance measures

Two main performance measures, namely the percentage of clients with QoS in the system (measuring the interactivity of the system), denoted as $pQoS$, and the server resource utilization⁶, i.e., bandwidth consumption, (measuring the cost of the algorithms), are of interest in the analysis. Results presented in this study are obtained by averaging the results of 50 simulation runs.

6.4 Summary

In this chapter, we have described our methodology to evaluating the proposed server placement approaches and client assignment algorithms. To obtain meaningful conclusions from the simulation results, we try to be as realistic as possible in modelling the networks and system workloads. This is achieved by using a wide range of network topologies, from synthetic to real ones, and implementing routing protocol that is widely used in the current Internet. In addition, different client distributions are also considered to further understand the performance of the proposed algorithms.

⁶Here, the resource utilization refers to that of the entire system.

Chapter 7

Results

In this chapter, we compare various server placement approaches and client assignment algorithms via simulations. To this end, we first investigate the impacts of server placement approaches and client assignment algorithms separately. Then, their combined impacts are investigated. More specifically, while studying the impacts of server placement approaches, we use the basic client assignment mechanism, i.e., the random assignment algorithm (RanZ/VirC). On the other hand, while studying the impacts of client assignment algorithms, the Random server placement approach is used.

7.1 Default settings and notations

Unless otherwise stated, the following assumptions and default values are used in the simulations. The clients are uniformly distributed in the physical world as well as in the virtual world. The number of servers, number of zones, number of clients, and the value of correlation parameter are 20, 400, 5000 and 0.5, respectively. The minimum bandwidth capacity of a server is set to 10Mbps, while the total bandwidth capacity of the system is set to 2500Mbps.

CHAPTER 7. RESULTS

The maximum round-trip delay between any two nodes in the network topology is set to 300ms. To simulate the well-provisioned inter-server connections (if necessary) which have lower delays than client-server connections, we set the network latency between any two geographically distributed servers to 50%¹ of the actual latency values obtained from the topology generator, as in [73]. The interactivity requirement, i.e., the DVE delay bound D is set to 150ms.

Different DVE configurations could be used for performance evaluation. A specific DVE configuration is determined by the number of servers, the number of zones, the number of clients and the total resource capacity of the system. We use the notation *number of servers-number of zones- number of clients-capacity* to denote a DVE configuration. For example, the notation 20s-400z-5000c-2500cp means that the DVE has 20 servers, 400 zones, 5000 clients and 2500Mbps server bandwidth in total.

7.2 Impacts of server placement approaches

Fig. 7.1 and Fig. 7.2 show the cumulative distribution function (CDF) of client-server round-trip communication delays under several server placement approaches and network topologies. For flat topologies (Fig. 7.1), we observe that the Max-router placement performs very well: in terms of $pQoS$, the performance improvements are 157% (F-W topology), 168% (F-BA topology) and 26% (F-R topology) compared to the Random placement, and 157%, 275% and 92% compared to the Min-router placement. Moreover, from these figures, it is clear that Max-router not only has a high ratio of clients with QoS, but also provide better interactivity for clients that are without QoS guarantees.

¹We refer to this value as the *delay reduction factor*.

CHAPTER 7. RESULTS

Figure 7.1: Impacts of server placement - CDF with flat topologies

It should be noted that the performance improvement in F-R topology is not as high as those in F-W and F-BA topologies. However, this is not because the Max-router placement performs poorly in F-R topology. As observed in Fig. 7.1(c), the Random placement also performs quite well in F-R topology. One possible explanation for this result is that the F-R topology, which is a real-world AS-level topology collected from [25], has good connectivity, i.e. any node can reach any other node with a small number of hop counts. Indeed, the average round-trip delay between any two nodes in F-R is much smaller than those in F-W and F-BA, about 30% and 57%, respectively.

For hierarchical topologies (Fig. 7.2), we obtain similar results to the case of flat topologies, except that the Max-AS/Max-router placement performs the best: its performance improvements in terms of $pQoS$ over the Random placement are 38% (H-BA/W)

CHAPTER 7. RESULTS

Figure 7.2: Impacts of server placement - CDF with hierarchical topologies

and 48% (H-R/R). This can be explained by the fact that Max-AS/Max-router spreads the servers across multiple ASes, while Max-router may place several servers in the same AS. In addition, it is observed that Max-AS/Min-router and Min-AS/Max-router both perform worse than Max-AS/Max-router, while Min-AS/Max-router is the worst among the three. This observation suggests that the AS-level node degree seems to be the major factor that affects the performance. However, we should also note that only selecting the ASes with largest AS-level degrees to place the servers is not enough: the router-level node degree within an AS is also an important factor.

Surprisingly, our results here are similar to the observations in [90], although [90] addresses the Web replica placement problem, which is very different from the server placement problem in DVEs. Recall that for a typical DVE, which is partitioned into multiple distinct zones, we do not know in advance which server a client will connect to. Thus, we have to place the servers without knowing clients' locations, then randomly assign zones to servers and let each client connect to the server hosting its zone. In contrast, in the context of Web replica placement problem, a client just simply connects to its closest server. Hence, to some extent, the Web replica placement problem may be

CHAPTER 7. RESULTS

“easier” to solve than the DVE server placement problem. Nonetheless, it’s clear that the degree-based placement approach is universal, i.e., it can be applied into both DVE server placement and Web replica placement with good results.

7.2.1 Impacts of the number of servers

Figure 7.3: Impacts of the number of servers on server placement approaches

Fig. 7.3 shows the impacts of server number on various server placement approaches². In this experiment, we vary the number of servers from 10 to 40 while keeping the same total system capacity. From Fig. 7.3, it seems that increasing number of servers does not have any positive effect on the interactivity, i.e., $pQoS$. In contrast, it may badly affect the $pQoS$, especially in the case of Max-router (Fig. 7.3(a)) and Max-AS/Max-router (Fig. 7.3(b)). Hence, we conclude that placing too many servers without increasing system capacity may not result in better interactivity for DVEs.

7.2.2 Impacts of server capacity

In this experiment, we fix the number of servers at 20, and increase the total system capacity from 1500Mbps to 3500Mbps. Fig. 7.4 shows the impacts of the system capacity.

²We only show the results for F-BA and H-BA/W topologies since results for other topologies are similar.

CHAPTER 7. RESULTS

Figure 7.4: Impacts of server capacity on server placement approaches

It seems that increasing capacity also does not have any impact on the interactivity. This and the above result (impact of server number) may suggest that it's hard to improve the “interactivity level” of a DVE, i.e., in case a DVE requires tight delay bound, using only degree-based server placement approaches. However, later in this chapter, we will show that it is possible to further enhance DVE's interactivity level by combining degree-based server placement with proper client assignment.

7.2.3 Impacts of correlations

Figure 7.5: Impacts of virtual world-physical world correlations on server placement approaches

Fig. 7.5 shows the impacts of physical world-virtual world correlation values. Recall

CHAPTER 7. RESULTS

that the higher the correlation value is, the stronger the tendency for clients from the close geographic locations to gather in specific zones of the virtual world. However, it seems that this phenomenon does not have any impact on $pQoS$ of all the server placement approaches.

7.2.4 Impacts of client distributions

Figure 7.6: Impacts of client distributions on server placement approaches - $pQoS$

Figure 7.7: Impacts of client distributions on server placement approaches - resource utilization

In this experiment, we evaluate the impacts of different client distributions in both the physical world and the virtual world on the performance of server placement approaches. Fig. 7.6 and Fig. 7.7 show the simulation results. It is observed that client

CHAPTER 7. RESULTS

distributions do have some impacts on $pQoS$ and resource utilization. More specifically, the performance in terms of $pQoS$ of all server placement approaches seems to be better when clients are clustered in the physical world (distribution type 1 and 3, Fig. 7.6). On the other hand, the clustered distribution of clients in the virtual world mainly has effects on the resource utilization (see Fig. 7.7). We see that the resource utilization becomes very high when clients are clustered in the virtual world (distribution type 2 and 3).

7.2.5 Summary

We have evaluated several degree-based server placement approaches to enhance DVE's interactivity via extensive simulations. As far as we know, this is the first work to consider server placement approaches in DVEs. Our main findings are as follows:

- The degree-based server placement approaches such as Max-router and Max-AS/Max-router, which do not require to know the clients' locations in advance, are able to perform very well in terms of interactivity, i.e., $pQoS$, compared to the Random server placement approach. For a wide range of network topologies that we have tested, the improvement in interactivity over the Random placement approach ranges from 26% to 168%. This result indicates that the degree-based server placement methodology is suitable for both the Web replica placement problem [90], and the server placement problem in DVEs.
- Common phenomena in large-scale DVEs like physical world-virtual world correlation, or client clustering do not seem to have bad effects on the interactivity provided by degree-based server placement approaches. In contrast, the interactivity seems to be better when clients are clustered in the network.
- It seems that increasing server number and capacity have negative or no effect

CHAPTER 7. RESULTS

on the interactivity level provided by the degree-based server placement approaches. This result suggests that it may be inappropriate to enhancing DVE's interactivity by just adding more servers or capacity, when only degree-based server placement is used. Moreover, the simulation results show that using a small number of servers may produce better interactivity than using many servers. However, using less servers may worsen the scalability of the system, since the resource becomes more concentrated. Hence, a carefully selected number of servers should strike a balance between scalability and interactivity. To further enhance interactivity, we shall rely on efficient client assignments, which will be investigated later in this chapter.

7.3 Impacts of client assignment algorithms

In investigating the proposed client assignment algorithms, we use two DVE configurations, 5s-30z-400c-200cp (small) and 20s-400z-5000c-2500cp (large). The small configuration is only used to enable the comparison between our algorithms and the optimal solution produced by the MILP solver `lp_solve` [14]. In all other cases, we use the large configuration. Note that to evaluate the impacts of the client assignment algorithms, in this section we use the Random server placement approach to place the servers.

7.3.1 “Normal” inter-server networks

In this section, we study the performance of client assignment algorithms in the case that the networks between servers are not well-provisioned, thus the word “normal”. Hence, the client assignment algorithms considered in this section include `RanZ/VirC`, `GreZ-1/VirC` and `GreZ-2/VirC`. Note that these algorithms do not incur inter-server

CHAPTER 7. RESULTS

Figure 7.8: Impacts of client assignment - CDF with flat topologies, small configuration, normal inter-server networks

communications.

Fig. 7.8, Fig. 7.9, Fig. 7.10 and Fig. 7.11 show the CDF of client-server round-trip communication delays for our algorithms in the small and large configurations with different network topologies. In all cases, we observe that the GreZ-1/VirC and GreZ-2/VirC significantly outperforms the delay-oblivious RanZ/VirC algorithm in terms of $pQoS$. We also observed that among the three algorithms (RanZ/VirC, GreZ-1/VirC and GreZ-2/VirC), GreZ-1/VirC has the best performance in terms of $pQoS$. This can be explained by the fact that GreZ-1/VirC aims to directly minimize the total number of clients without QoS in the system, while GreZ-2/VirC aims to assign a zone to a server that is close (in terms of network delay) to the center of mass of the client population in

CHAPTER 7. RESULTS

Figure 7.9: Impacts of client assignment - CDF with hierarchical topologies, small configuration, normal inter-server networks

Figure 7.10: Impacts of client assignment - CDF with hierarchical topologies, large configuration, normal inter-server networks

that zone. The performance improvements of GreZ-1/VirC over RanZ/VirC are 63% (F-W), 58% (F-BA), 16% (F-R), 35% (H-BA/W) and 25% (H-R/R) for small configuration, and 116% (F-W), 110% (F-BA), 26% (F-R), 48% (H-BA/W) and 38% (H-R/R) for large configuration.

We also note that the $pQoS$ values of GreZ-1/VirC are close to the optimal results given by the branch-and-bound algorithm implemented in the lp_solve software. Note that lp_solve can only be applied to small size DVEs, i.e., the small configuration. The average execution time of lp_solve for this configuration is around 1 second. For the large configuration, the results by lp_solve are not shown since the execution time was too

CHAPTER 7. RESULTS

Figure 7.11: Impacts of client assignment - CDF with flat topologies, large configuration, normal inter-server networks

long (not finished after more than 10 hours), which is clearly impractical for interactive application like DVEs which need timely assignment decisions. In both small and large configurations, all of our proposed algorithms took less than 0.5 second to execute.

Impacts of the number of servers

Fig. 7.12 shows the impacts of server number on the proposed client assignment algorithms when we fix the total system capacity. It is observed that the $pQoS$ values of both GreZ-1/VirC and GreZ-2/VirC are slightly increased, while the $pQoS$ of RanZ/VirC seems invariant when the number of servers increases. Although the increases are small in GreZ-1/VirC and GreZ-2/VirC, these results are different from those of server placement approaches, where the $pQoS$ values of the best placement approaches decrease when

CHAPTER 7. RESULTS

Figure 7.12: Impacts of the number of servers on client assignment algorithms, normal inter-server networks

the number of servers increases. Hence, the results here may suggest that when using GreZ-1/VirC or GreZ-2/VirC, we can add more servers to the system to strengthen its scalability without worrying that the interactivity will decrease as in the case of using only server placement approaches.

Impacts of server capacity

Figure 7.13: Impacts of server capacity on client assignment algorithms, normal inter-server networks

Fig. 7.13 shows the impacts of the system capacity. It is observed that the $pQoS$ values of GreZ-1/VirC and GreZ-2/VirC increase fairly well when the system capacity increases. For instance, in Fig. 7.13(a), when the capacity increases from 1500Mbps to

CHAPTER 7. RESULTS

2500Mbps, the increments in $pQoS$ values of GreZ-1/VirC and GreZ-2/VirC are 32.2% and 34.5%, respectively. This result, which is in contrast with the results obtained for the server placement approaches, may suggest that we can achieve better interactivity by increasing system capacity when appropriate client assignment algorithms are used.

Impacts of correlations

Figure 7.14: Impacts of physical - virtual world correlation on client assignment algorithms, normal inter-server networks

The performance of our proposed algorithms with different physical world-virtual world correlation values is shown in Fig. 7.14. It is observed that the $pQoS$ values of both GreZ-1/VirC and GreZ-2/VirC increase significantly with the correlation value, while the result of the delay-oblivious algorithm RanZ/VirC does not change much. This demonstrates the effectiveness of the greedy algorithms when the clients in each zone of the virtual world are close to each other in the physical world. In all cases, GreZ-1/VirC is still the best algorithm in terms of $pQoS$. However, it is also noted that GreZ-2/VirC is a good alternative when the correlation value is very high.

CHAPTER 7. RESULTS

Impacts of client distributions

The simulation results for all assignment algorithms with different client distributions are shown in Fig. 7.15 and Fig. 7.16. In all cases, both GreZ-1/VirC and GreZ-2/VirC are always better than RanZ/VirC, regardless of the client distributions, while GreZ-1/VirC is the best algorithm.

Figure 7.15: Impacts of client distributions on client assignment algorithms - $pQoS$, normal inter-server networks

Figure 7.16: Impacts of client distributions on client assignment algorithms - resource utilization, normal inter-server networks

The impacts of client distributions on $pQoS$ of all algorithms are clear: For the greedy algorithms, the dominant factor seems to be the client distribution in the virtual world, for example, the $pQoS$ is better when the distribution type is 0 and 1, i.e., clients are

CHAPTER 7. RESULTS

distributed uniformly in the virtual world. For all distribution types (0, 1, 2, 3), if the client distributions in the virtual world of any two distribution types are the same, then a clustered distribution of clients in the network seems to result in a better $pQoS$.³ These observations can be explained if we look at Fig. 7.16, which shows the resource utilization for each distribution type. The distribution type 0 and type 1 use much less resource than type 2 and type 3, and if we recall that the system capacity may have great impacts on the $pQoS$ of the greedy algorithms, it's easy to see why GreZ-1/VirC and GreZ-2/VirC perform better if clients are not highly clustered in the virtual world.

For the delay-oblivious algorithm RanZ/VirC, since its performance is not affected by the system capacity, we see that its $pQoS$ almost remains the same regardless of the client distributions in the virtual world.

Impacts of imperfect inputs

The simulation results obtained above for the client assignment algorithms are based on the assumption that we have perfect information about the network delays between clients and servers. In practice, we usually have rough estimations of network delays rather than perfectly accurate information. To simulate the estimation error, we use the method presented in [93]. More specifically, we apply an error factor e to the perfect input data, i.e., assuming the exact value of the delay is d , then the delay value input to our algorithms is uniformly distributed in the range $[\frac{d}{e}, de]$. Note that although the algorithms use inaccurate inputs to make assignment decisions, their resulted performances, i.e., $pQoS$, are still calculated based on the actual network delays after the assignment is done. As in [93], we use three different values of e : 1.2, 2 and 4, representing the

³This observation is similar to that in the server placement approaches.

CHAPTER 7. RESULTS

Figure 7.17: Impacts of imperfect inputs - F-BA topology, normal inter-server networks inaccuracies of the popular network delay estimation tools King [56] and IDMaps [53].

Table 7.1: $pQoS$ with imperfect input data, F-BA topology, normal inter-server networks

Error factor	RanZ/VirC	GreZ-1/VirC	GreZ-2/VirC
0	0.31	0.65	0.62
1.2	0.3	0.6	0.59
2	0.3	0.5	0.54
4	0.31	0.43	0.49

Fig. 7.17, 7.18, and Table 7.1, 7.2 show the simulation results obtained with the inaccurate estimations of network delays. We also include the case of accurate input data, i.e., $e = 0$ for comparisons. Despite the imperfect knowledge in network delay, the greedy algorithms are still much better than the random algorithm which does not consider network delays. In particular, for $e = 1.2$, GreZ-1/VirC is still the best algorithm

CHAPTER 7. RESULTS

Figure 7.18: Impacts of imperfect inputs - H-BA/W topology, normal inter-server networks

in terms of interactivity, and its $pQoS$ only decreases slightly compared to the cases that use perfect information, as observed in Table 7.1 and 7.2.

When the estimation error becomes large, i.e., $e = 2$ or 4 , we see that GreZ-2/VirC is the best algorithm. This can be explained by its low sensitivity to data errors, as observed in the last column of Table 7.1 and 7.2. The performance of GreZ-2/VirC decreases rather slightly even when the estimation error is very large, i.e., $e = 4$, compared to the case of perfect input data. This suggests that GreZ-2/VirC is a more suitable choice for the client assignment problem under Internet environments, where perfectly accurate input data for assignment algorithms is usually impossible or very costly to obtain.

CHAPTER 7. RESULTS

Table 7.2: $pQoS$ with imperfect input data, H-BA/W topology, normal inter-server networks

Error factor	RanZ/VirC	GreZ-1/VirC	GreZ-2/VirC
0	0.63	0.94	0.89
1.2	0.64	0.9	0.88
2	0.63	0.85	0.87
4	0.64	0.8	0.85

Table 7.3: Impacts of client assignments on $pQoS$, well-provisioned inter-server networks, small configuration

Algo. vs. Topo.	F-W	F-BA	F-R	H-BA/W	H-R/R
RanZ/CloC (RanZ/VirC)	0.24 (0.28)	0.26 (0.32)	0.66 (0.8)	0.58 (0.63)	0.65 (0.72)
GreZ-1/CloC (GreZ-1/VirC)	0.39 (0.47)	0.43 (0.48)	0.8 (0.91)	0.71 (0.83)	0.75 (0.89)
GreZ-2/CloC (GreZ-2/VirC)	0.4 (0.43)	0.39 (0.45)	0.77 (0.9)	0.73 (0.8)	0.76 (0.86)
RanZ/GreC	0.28	0.33	0.81	0.71	0.79
GreZ-1/GreC	0.48	0.49	0.93	0.85	0.91
GreZ-2/GreC	0.44	0.46	0.91	0.82	0.87
lp_solve	0.49	0.51	0.94	0.86	0.92

7.3.2 Well-provisioned inter-server networks

In this section, we study the performance of client assignment algorithms in the case that the networks between servers are well-provisioned. As mentioned in Chapter 5 (Client assignment algorithms), the client assignment algorithms considered in this case include RanZ/CloC, GreZ-1/CloC, GreZ-2/CloC, RanZ/GreC, GreZ-1/GreC and GreZ-2/GreC. To simulate the well-provisioned inter-server networks, we reduce the network latency between any two geographically distributed servers to 50% of the actual latency values obtained from the topology generator, as in [73].

Table 7.3, 7.4, and Table 7.5, 7.6 show the $pQoS$ and resource utilization, while Fig.

CHAPTER 7. RESULTS

Table 7.4: Impacts of client assignments on resource utilization, well-provisioned inter-server networks, small configuration

Algo. vs. Topo.	F-W	F-BA	F-R	H-BA/W	H-R/R
RanZ/CloC (RanZ/VirC)	0.99 (0.61)	0.99 (0.61)	0.95 (0.61)	0.99 (0.61)	0.99 (0.61)
GreZ-1/CloC (GreZ-1/VirC)	0.97 (0.61)	0.95 (0.61)	0.89 (0.61)	0.97 (0.61)	0.97 (0.61)
GreZ-2/CloC (GreZ-2/VirC)	0.97 (0.61)	0.95 (0.61)	0.9 (0.61)	0.95 (0.61)	0.96 (0.61)
RanZ/GreC	0.7	0.71	0.68	0.8	0.78
GreZ-1/GreC	0.64	0.64	0.62	0.66	0.65
GreZ-2/GreC	0.63	0.63	0.62	0.67	0.66
lp_solve	0.95	0.92	0.65	0.71	0.67

Table 7.5: Impacts of client assignments on $pQoS$, well-provisioned inter-server networks, large configuration

Algo. vs. Topo.	F-W	F-BA	F-R	H-BA/W	H-R/R
RanZ/CloC (RanZ/VirC)	0.24 (0.29)	0.25 (0.31)	0.67 (0.77)	0.63 (0.64)	0.7 (0.71)
GreZ-1/CloC (GreZ-1/VirC)	0.50 (0.65)	0.51 (0.64)	0.76 (0.96)	0.79 (0.94)	0.81 (0.97)
GreZ-2/CloC (GreZ-2/VirC)	0.51 (0.61)	0.51 (0.61)	0.78 (0.94)	0.80 (0.88)	0.82 (0.91)
RanZ/GreC	0.34	0.35	0.82	0.81	0.89
GreZ-1/GreC	0.66	0.65	0.97	0.97	0.99
GreZ-2/GreC	0.63	0.62	0.95	0.93	0.96

(a) F-BA topology

(b) H-BA/W topology

Figure 7.19: Impacts of client assignment algorithms - CDF, well-provisioned inter-server networks, small configuration

CHAPTER 7. RESULTS

Table 7.6: Impacts of client assignments on resource utilization, well-provisioned inter-server networks, large configuration

Algo. vs. Topo.	F-W	F-BA	F-R	H-BA/W	H-R/R
RanZ/CloC (RanZ/VirC)	0.99 (0.58)	0.99 (0.58)	0.99 (0.58)	0.98 (0.58)	0.99 (0.58)
GreZ-1/CloC (GreZ-1/VirC)	0.99 (0.58)	0.93 (0.58)	0.93 (0.58)	0.99 (0.58)	0.98 (0.58)
GreZ-2/CloC (GreZ-2/VirC)	0.99 (0.58)	0.9 (0.58)	0.9 (0.58)	0.97 (0.58)	0.99 (0.58)
RanZ/GreC	0.84	0.74	0.74	0.91	0.88
GreZ-1/GreC	0.66	0.59	0.58	0.62	0.6
GreZ-2/GreC	0.64	0.59	0.59	0.67	0.66

7.19 and Fig. 7.20 show the CDFs of communication delays for the proposed client assignment algorithms for both small and large configurations with different network topologies. The first observation is that all the algorithms that have CloC as the refined assignment perform badly, more specifically, even worse (in terms of both $pQoS$ and resource utilization) than the algorithms that have VirC as the refined assignment,⁴ although CloC's principle is widely used (and produces good results) in the context of Web replica placement problem. Note that, for comparison, the performances of algorithms with the refined assignment VirC are also included in Table 7.3, 7.4, and Table 7.5, 7.6 (the values inside the brackets of the first 3 rows of each table).

We believe that the main reason for the ineffectiveness of CloC when applying to DVEs is that the network path from a client to its target server via that client's closest server may not be the shortest paths in many cases, despite the inter-server networks are well-provisioned. Hence, selecting the closest servers as contact servers for clients may result in high communication delays, and incur much more resource consumption. This

⁴Algorithms with VirC as the refined assignment do not require well-provisioned inter-server networks.

CHAPTER 7. RESULTS

Figure 7.20: Impacts of client assignment algorithms - CDF, well-provisioned inter-server networks, large configuration

clearly illustrates the difference between the client assignment problem in DVEs and in Web environments: the “target server” of a client in a DVE may not be its closest server, but this is true for the latter.

We have also tested CloC with higher delay reduction factors such as 70% or 90% as suggested in [73]. However, the final conclusion is that in most cases, CloC *worsens* the $pQoS$ obtained from the initial assignment, and significantly increases the resource utilization. Hence, we deem it inappropriate to solve the client assignment problem in DVEs, and decide not to further investigate it in this study. From this point, for well-provisioned inter-server networks, we only consider the following greedy algorithms: RanZ/GreC, GreZ-1/GreC and GreZ-2/GreC.

Impacts of the number of servers, capacity, correlations and client distributions

In this section, we repeat the experiments that were conducted in the case of normal inter-server networks, and obtain similar results. Hence, for clarity, the most representative results are presented here.

CHAPTER 7. RESULTS

Figure 7.21: Impacts of the number of servers on client assignment algorithms - H-BA/W topology, well-provisioned inter-server networks

Figure 7.22: Impacts of server capacity on client assignment algorithms - H-BA/W topology, well-provisioned inter-server networks

Fig. 7.21, 7.22 and 7.23 show the impacts of server number, capacity and physical world-virtual world correlation, respectively. We observe that the $pQoS$ values for all algorithms (RanZ/GreC, GreZ-1/GreC and GreZ-2/GreC) are improved when these parameters increase, except that the correlation value does not seem to have any considerable impact on RanZ/GreC. On the other hand, when the correlation value is very high, the performances of GreZ-1/GreC and GreZ-2/GreC are comparable in terms of both $pQoS$ and resource utilization.

Similar to the case of normal inter-server connection, client distributions do have some impacts on the performance of the client assignment algorithms, however GreZ-1/GreC

CHAPTER 7. RESULTS

Figure 7.23: Impacts of physical world-virtual world correlations on client assignment algorithms - H-BA/W topology, well-provisioned inter-server networks

Figure 7.24: Impacts of client distributions on client assignment algorithms - H-BA/W topology, well-provisioned inter-server networks

is always the best algorithms regardless of client distributions. This time, the client distribution in the virtual world still seems to be the dominant factor for the observed performances, as shown in Fig. 7.24. In addition, it seems that the client distribution in the network does not have a significant impact on the interactivity of GreZ-1/GreC and GreZ-2/GreC. This observation is different from the case of normal inter-server networks. One possible explanation may be due to the fact that greedy assignment algorithms (GreZ-1, GreZ-2, and GreC) are used in both the initial assignment and the refined assignment. Such combination may have taken the advantage of well-provisioned inter-server networks to reduce the communication delay regardless of the client distributions

CHAPTER 7. RESULTS

in the network.

Impacts of inter-server network delays

Figure 7.25: Impacts of inter-server network delays on client assignment algorithms - H-BA/W topology

Fig. 7.25 shows the impacts of the delay reduction factor on our assignment algorithms. Note that when the reduction factor is 0, i.e., the inter-server networks are not well-provisioned, GreC is effectively equal to VirC. One important observation here is, unlike RanZ/GreC, algorithm GreZ-1/GreC and GreZ-2/GreC do not heavily depend on the reduction factor. This result suggests that GreZ-1/GreC and GreZ-2/GreC are still able to perform well even in the case there's little delay reduction in the inter-server networks.⁵ This is because the initial greedy assignment phase already provides a good value of $pQoS$, and there may be not much room for further improvement in the refined assignment. Nevertheless, whenever low-delay inter-server networks are available, the refined assignment phase is still beneficial.

⁵We should note that providing low-delay networks that interconnect geographically distributed servers may be expensive, especially in the current Internet where well-known network QoS techniques like IntServ or DiffServ have not been widely deployed. Nevertheless, without a dedicated, private high-speed network, we still can reduce the communication delays between distributed servers, by closely monitoring the routing paths among these servers [31].

CHAPTER 7. RESULTS

Impacts of imperfect inputs

Table 7.7: $pQoS$ with imperfect input data, H-BA/W topology, well-provisioned inter-server networks

Error factor	1.2	2	4
RanZ/GreC	0.75	0.6	0.54
GreZ-1/GreC	0.93	0.8	0.65
GreZ-2/GreC	0.91	0.81	0.65
RanZ/VirC	0.64	0.64	0.64
GreZ-1/VirC	0.9	0.85	0.8
GreZ-2/VirC	0.88	0.87	0.85

Table 7.8: Resource utilization with imperfect input data, H-BA/W topology, well-provisioned inter-server networks

Error factor	1.2	2	4
RanZ/GreC	0.94	1	1
GreZ-1/GreC	0.65	0.8	1
GreZ-2/GreC	0.7	0.83	1
RanZ/VirC	0.58	0.58	0.58
GreZ-1/VirC	0.58	0.58	0.58
GreZ-2/VirC	0.58	0.58	0.58

We evaluate the performances of RanZ/GreC, GreZ-1/GreC and GreZ-2/GreC with imperfect input data. Similar to the previous experiment, we use three different values of the error factor e : 1.2, 2 and 4, to add random noise to the network delays. The simulation results are shown in Table 7.7, 7.8, and Fig. 7.26. Note that we also include the results for RanZ/VirC, GreZ-1/VirC and GreZ-2/VirC (the algorithms proposed for normal inter-server networks) for comparison.

From the simulation results, we see that when the delay estimation error is small, i.e., $e = 1.2$, all the algorithms that use the knowledge of network delays are still better than the delay-oblivious algorithm RanZ/VirC in terms of $pQoS$. However, when the

CHAPTER 7. RESULTS

Figure 7.26: Impacts of imperfect inputs - H-BA/W topology, well-provisioned inter-server networks

error becomes larger, i.e., $e = 2$, RanZ/GreC performs even worse than RanZ/VirC in terms of both $pQoS$ and resource utilization. In this case, although GreZ-1/GreC and GreZ-2/GreC are still better than RanZ/VirC, they are not as good as GreZ-1/VirC and GreZ-2/VirC. Moreover, when e is very large, i.e., $e = 4$, GreZ-1/GreC and GreZ-2/GreC perform comparably to RanZ/VirC, which is much worse than GreZ-1/VirC and GreZ-2/VirC. This important observation suggests that the refined assignment algorithm GreC is quite sensitive to the estimation error. Hence, GreC should not be used when the network delays input to the assignment algorithms are highly inaccurate. In such scenario, GreZ-2/VirC is the best choice, since it not only exhibits low-sensitivity to delay estimation errors, but also does not require well-provisioned inter-server networks

CHAPTER 7. RESULTS

and thus incurs less resource utilization.

7.3.3 Summary

In this section, we have evaluated several client assignment algorithms in two cases: “normal” inter-server networks and well-provisioned inter-server networks. To observe the impacts of client assignments, we use the Random server placement approach. Our main findings in this regard are as follow:

- All greedy client assignment algorithms that utilize the knowledge of network delays outperform a delay-oblivious algorithm, although the delays input to these algorithms are inaccurate. However, when the delay estimation error becomes very large, the GreZ-2/VirC algorithm performs the best. Moreover, GreZ-2/VirC does not require well-provisioned inter-server networks. This property is of great significance for practical deployments, since usually an Internet-wide well-provisioned network is quite expensive.
- We note that unlike the case of server placement approaches, increasing server capacity, server number and physical world-virtual world correlation value do have positive impacts on the greedy client assignment algorithms, i.e., the $pQoS$ values of these algorithms are better. This suggests that we can enhance the interactivity level of DVEs by adding more resources to the system. In addition, client distributions also have some impacts on performance, for example, if the clients are uniformly distributed in the virtual world, the $pQoS$ values of the greedy assignment algorithms are better compared to the case that the clients are highly clustered in the virtual world. However, the greedy algorithms always outperform the delay-oblivious algorithm regardless of the client distributions in the virtual world or in the networks.

CHAPTER 7. RESULTS

- The performance improvements of good client assignment algorithms may seem not as impressive as the improvements obtained by good server placement approaches. For example, recall that for a typical DVE configuration (20s-400z-5000c-2500cp), the improvements of Max-router or Max-AS/Max-router over the Random placement approach in terms of $pQoS$ for all tested network topologies range from 26% to 168%. At the same time, the improvements of GreZ-1/VirC over RanZ/VirC is about 26% to 116%. However, we should note that good server placement approaches assume that we can place the servers at any location that we want. In practice, this sometimes may not be achievable, due to economic reasons or technical reason, e.g., the routers with high node degree may be very busy and thus it may be impossible to add any server to them [90]. In contrast, client assignment algorithms do not rely on such assumption, thus they can be applied right away at the user's wish.

7.4 Combined impacts of server placement and client assignment

In this section, we investigate the possibility of combining server placement approaches and client assignment algorithms to further enhance the interactivity of DVEs. We propose to study two cases, the first one is the case in which the DVE requires tighter delay bound than before, e.g., $100ms$, while the second case considers the scenario in which we are forced to place servers at bad locations, i.e., the routers with lowest node degrees. The reason for studying the first case is that we would like to know to which extent a combination of good server placement and good client assignment can enhance the interactivity of DVEs. For the second case, our aim is to answer the question whether

CHAPTER 7. RESULTS

it is possible for good client assignment algorithms to lift up the interactivity level of DVEs when the servers are already placed at bad network locations.

7.4.1 Tight delay bound

Figure 7.27: Combined impact - tight delay bound, flat topologies

We study the combined impact of server placement and client assignment using a higher interactivity requirement, i.e., we set the delay bound to $100ms$. In this experiment, we use a combination of Max-router (or Max-AS/Max-router)⁶ and GreZ-2/VirC.⁷

Fig. 7.27 and 7.28 show the CDF of communication delays for different network

⁶If hierarchical network topologies are used, the best server placement approach is Max-AS/Max-router. Otherwise, if flat topologies are used, Max-router is the best server placement approach.

⁷We select GreZ-2/VirC since it gives the best performances in terms of both $pQoS$ and resource utilization under Internet environments, where input data to client assignment algorithms are usually inaccurate.

CHAPTER 7. RESULTS

Figure 7.28: Combined impact - tight delay bound, hierarchical topologies

topologies. From these figures, it's clear that in all cases the combined approach Max-router (Max-AS/Max-router) + GreZ-2/VirC outperforms the Max-router (Max-AS/Max-router) or GreZ-2/VirC alone. More specifically, the improvements in terms of $pQoS$ of the combined approach over Max-router (Max-AS/Max-router) are 144% (F-W), 130% (F-BA), 29% (F-R), 90% (H-BA/W) and 59% (H-R/R). Similarly, the improvements over GreZ-2/VirC are 144% (F-W), 225% (F-BA), 62% (F-R), 34% (H-BA/W) and 33% (H-R/R). This observation suggests that combining good server placement and good client assignment is a very useful approach to further enhance the interactivity level of DVEs.

7.4.2 Bad server placement

As already mentioned, due to technical or economic constraints, sometimes we may not be able to place the servers at good network locations, i.e., the routers with high node degrees. Hence, we may be forced to place the servers at some “bad” locations. In this experiment, we would like to know whether good client assignment algorithms can still improve the interactivity in the case servers are all placed at the routers with lowest node degrees.

CHAPTER 7. RESULTS

Figure 7.29: Combined impact - bad server placement, flat topologies

Fig. 7.29 and 7.30 show the results of this experiment with different network topologies. It is observed that, despite the bad server placement, the GreZ-2/VirC can still improve the interactivity of the system significantly. More specifically, the improvements of Min-router + GreZ-2/VirC over Min-router + RanZ/VirC in terms of $pQoS$ are 150% (F-W), 100% (F-BA), 44% (F-R), 42% (H-BA/W) and 28% (H-R/R).

7.5 Summary

In this chapter, we have evaluated several degree-based server placement approaches and client assignment algorithms. Appropriate server placements and client assignments perform very well in terms of enhancing interactivity for DVEs. Degree-based server

CHAPTER 7. RESULTS

Figure 7.30: Combined impact - bad server placement, hierarchical topologies

placement approaches, which are adapted from well-known Web replica placement mechanisms, do not require to know the clients' locations in advance, thus they do not suffer from the inaccuracies in network delay estimations, and can be applied straightforwardly to DVEs. However, these approaches may not be applicable in every scenarios, due to some inherent technical and economic constraints.

On the other hand, client assignment algorithms use the knowledge of network delays to determine appropriate client-to-server assignment. This approach is universally applicable, since it does not require any special server positioning. However, the performance of client assignment algorithms seems to inevitably depend on accurate information of the client-server network delays. However, our experiment results have shown that a greedy assignment algorithm, GreZ-2/VirC, exhibits low sensitivity to delay estimation errors, even in the case the error is very large. This property is extremely useful, since the assignment algorithms have to work under Internet environments, where perfectly accurate input data for assignment algorithms is usually impossible or very costly to obtain.

We have also studied the combined impacts of server placement and client assignment.

CHAPTER 7. RESULTS

There are several situations that we need to investigate such combination, e.g., improving the interactivity level of the DVE when the delay bound is very tight, or dealing with a bad server placement. The experimental results showed that a combination of good server placement and good client assignment is able to significantly improve the interactivity of the DVE, compared to the cases in which only server placement or client assignment is used. Moreover, a good client assignment can still enhance the interactivity level of the DVE even when servers are already located to inappropriate network locations.

Chapter 8

Conclusions and future directions

Large-scale DVEs refer to a class of applications that enable thousands of geographically distributed users to communicate and interact simultaneously with each other in a synthetic, computer-generated 3D virtual world. In such applications, enhancing interactivity is of crucial importance, since DVEs are human-in-the-loop applications, and thus their success or failure greatly depends on the users' perceptions. More specifically, the users may not feel like they are interacting with other elements in the virtual world if the responses from the DVE are much slower than what the users experience in real life. Hence, in this thesis, our ultimate goal is to develop efficient mechanisms to enhance the interactivity of DVEs.

However, seeking a good balance between enhancing interactivity and other important issues in DVEs such as maintaining consistency, managing resource bottlenecks and securing the virtual world system under various technical problems such as large Internet latency, high resource demands and high potential of security threats is a very challenging task. To this end, traditional application-centric approaches, e.g., dead reckoning, usually aim to “mask” the effect of network latency via predicting the states of remote

CHAPTER 8. CONCLUSIONS AND FUTURE DIRECTIONS

objects. Since the activities of human users in DVEs are highly unpredictable, such approach may result in visual perception errors, or in more serious cases, inconsistent DVE states.

In this thesis, we have proposed an approach which is based on a multiple-server networking architecture to enhance the interactivity of DVEs by trying to *directly* minimize the network latencies in client-server communications. To this end, we have developed and evaluated several network-based mechanisms to effectively achieve our goal. In this chapter, we summarize the contributions of this thesis, and suggest possible extensions to our work.

8.1 Contributions

The main contributions of this thesis are as follows.

- We have proposed a network-centric methodology for enhancing DVE interactivity via *directly reducing network latencies in communications between DVE's elements*. Our methodology is different from traditional application-centric approaches, e.g., dead reckoning, which usually try to “hide” the effect of network latencies by predicting the state of remote users. Since the activities of human users in DVEs are highly unpredictable, such approach may result in visual perception errors, or in more serious cases, inconsistent DVE states.

- We have reviewed and evaluated several *networking architectures* for DVEs. Networking architecture is very important in DVEs, since it defines how different entities in the system can communicate with each other. By carefully assessing the pros and cons of each architecture, the multiple-server architecture is selected. This architecture

CHAPTER 8. CONCLUSIONS AND FUTURE DIRECTIONS

essentially provides good ways to implement consistency control, resource management and security techniques.

- We have discussed and evaluated several *partitioning approaches* for DVEs. To handle large-scale DVEs with thousands of simultaneous users, it is essential to partition the huge virtual world into multiple smaller zones to distribute it among the servers. To this end, through extensive analysis and real implementation using the HLA RTI as the middleware, the zone-based, discrete partitioning approach is selected.

- We have proposed to consider the *server placement* problem to reduce the client-server communication delays in DVEs. Existing approaches for Web replica placements are not directly suitable for DVEs, as the natures of the problems are different. Hence, we have adapted and proposed several server placement approaches for DVEs. These approaches are referred to as degree-based server placements, i.e., they aim to place servers at the network nodes (routers) that have high node degrees. Degree-based placement approaches do not require to know clients' locations in advance, thus they can be applied directly to DVEs. To our knowledge, this is the first work to consider the server placement problem for DVEs.

- Degree-based server placement approaches sometimes cannot be applied due to technical and economic constraints, and it's hard to further enhance the interactivity level of the DVE if needed, e.g., the DVE requires a very tight delay bound, using only server placement approaches. Hence, independent of any specific server placement, we have proposed and formulated a new problem, termed the *client assignment* problem. This problem essentially concerns how to assign clients to distributed servers to reduce the client-server communication delays. A two-phase approach, which consists of an

CHAPTER 8. CONCLUSIONS AND FUTURE DIRECTIONS

initial assignment and a refined assignment, is proposed to solve this problem. Each phase is then proved to be NP-hard. Therefore, we have developed several simple yet effective two-phase algorithms to address the client assignment problem.

- To evaluate the proposed approaches and algorithms, we have built a realistic simulation model, in which a wide range of network topologies are used. We have also implemented real-world routing protocols, and modelled network delays according to real Internet measurements. Various client distributions and correlations have been used to simulate different scenarios in real applications.

- We have conducted extensive simulations to evaluate the effectiveness of the proposed server placement approaches and client assignment algorithms. Our main findings are as follows. Degree-based server placement approaches such as Max-router or Max-AS/Max-router significantly outperform the Random placement approach, while most of the proposed greedy assignment algorithms outperform the delay-oblivious algorithm. Moreover, the simulation results have shown that the GreZ-2/VirC algorithm exhibits low sensitivity to the inaccuracies in input data, even when these inaccuracies are very large. This property is of great importance for practical deployments, since we usually do not have perfectly accurate input data for the assignment algorithms in Internet environments.

We have also evaluated the combined impacts of server placement and client assignment. The results showed that a combination of good server placement and good client assignment can greatly enhance the interactivity level of the DVE. In addition, a good client assignment can still be of benefit even when the servers are already placed into some bad network locations, e.g., routers with low node degrees.

CHAPTER 8. CONCLUSIONS AND FUTURE DIRECTIONS

8.2 Future directions

We have obtained several important results in investigating network-based mechanisms to enhance the interactivity of DVEs. However, there are still some open issues in this research. For further study, we suggest the following directions.

- Our client assignment algorithms are centralized. This property facilitates management and easy deployments for practical use, however the overhead may be high for a central server. Distributed algorithms for client assignment is certainly a good research direction to complement our current work.

- In this study, we have assumed that the servers have sufficient computational power, and thus servers' processing delays, e.g., input processing, message sending, etc. are negligible compared to the Internet delay. It would be interesting if one can propose a model that incorporate both network delay and processing delay.

- Finally, if possible, we would like to further investigate the performance of the degree-based server placements and client assignments in the real deployment of a large-scale DVE, such as an MMOG. In this regard, it would be very interesting to study the side effects on the network caused by placing servers at high degree routers, and to conduct subjective evaluation to understand the interactivity improvements of our mechanisms on human users.

Bibliography

- [1] *Otter: Tool for Topology Display*, <http://www.caida.org/tools/visualization/otter>.
- [2] *A compendium of NP optimization problems*, www.nada.kth.se/~viggo/problemelist/.
- [3] *Akamai*, <http://www.akamai.com>.
- [4] *American's Army: the official U.S. Army's game*, <http://www.americasarmy.com/>.
- [5] *Autonomous System*, http://en.wikipedia.org/wiki/Autonomous_system.
- [6] *Bin Packing problem*, http://en.wikipedia.org/wiki/Bin_packing.
- [7] *Branch and bound method*, <http://www.nist.gov/dads/HTML/branchNbound.html>.
- [8] *BRITE Internet Topology Generator*, <http://www.cs.bu.edu/brite>.
- [9] *Content Delivery Networks*, <http://en.wikipedia.org/wiki/CDN>.
- [10] *DMSO HLA Runtime Infrastructure*, <https://www.dmsso.mil/>.
- [11] *Evequest*, <http://www.everquest.com>.
- [12] *Exodus*, <http://www.exodus.com>.
- [13] *Microsoft's Age of Empires*, <http://www.microsoft.com/games/empires/>.

BIBLIOGRAPHY

- [14] *Mixed Integer Linear Programming solver lp_solve*,
http://groups.yahoo.com/group/lp_solve.
- [15] *Multihoming*, <http://en.wikipedia.org/wiki/Multi-homed>.
- [16] *Netgames 2005*, <http://www.research.ibm.com/netgames2005/>.
- [17] *OMNeT++*, <http://www.omnetpp.org>.
- [18] *OpenSkies MMOG Framework*, <http://www.openskies.net/>.
- [19] *Quicksort algorithm*, <http://en.wikipedia.org/wiki/Quicksort>.
- [20] *Route Views project*, <http://www.routeviews.org>.
- [21] *Spacewar*, <http://www3.sympatico.ca/maury/games/space/spacewar.html>.
- [22] *SSF: Gallery of Network Models*, <http://www.ssfnet.org/Exchange/gallery/>.
- [23] *The DIVE project*, <http://www.sics.se/dive/>.
- [24] *The Network Simulator ns-2*, <http://www.isi.edu/nsnam/ns>.
- [25] *The NLANR project*, <http://moat.nlanr.net/Routing/rawdata>.
- [26] *The PARADISE project*, <http://www-dsg.stanford.edu/paradise.html>.
- [27] *Wikipedia - the free encyclopedia*, <http://en.wikipedia.org>.
- [28] *IEEE standard for distributed interactive simulation - application protocols*, IEEE Std 1278.1a-1998 (1998).

BIBLIOGRAPHY

- [29] A. Abdelkhalek, A. Bilas, and A. Moshovos, *Behavior and Performance of Interactive Multi-player Game Servers*, Special Issue of Cluster Computing: the Journal of Networks, Software Tools and Applications (2002).
- [30] H. Abrams, K. Watsen, and M. Zyda, *Three-tiered Interest Management for Large-Scale Virtual Environments*, Proc. of the ACM Symposium on Virtual Reality Software and Technology (VRST), 1998.
- [31] D. G. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, *Resilient overlay networks*, Proc. of the 18th ACM Symposium on Operating Systems Principles, 2001.
- [32] L. R. Bahl and D. T. Tang, *Optimization of concentrator locations in teleprocessing networks*, Proc. Symp. on Computer Communications and Teletraffic, 1972, pp. 355–362.
- [33] A.L. Barabasi and R. Albert, *Emergence of Scaling in Random Networks*, Science (1999), 509–512.
- [34] D. Bauer, S. Rooney, and P. Scotton, *Network infrastructure for massively distributed games*, Proc. of NetGames, 2002.
- [35] J. Beardsley, *Massively multiplayer game development*, ch. Seamless Servers: the Case for and Against, pp. 213–227, Charles River Media, 2003.
- [36] Blizzard, *Worlds of Warcraft*, <http://www.worldofwarcraft.com>.
- [37] M. Capps, *NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments*, IEEE Computer Graphics and Applications **20** (2000).

BIBLIOGRAPHY

- [38] G. Carneiro and C. Arabe, *Load Balancing for Distributed Virtual Reality Systems*, Proc. of the International Symposium on Computer Graphics, Image Processing, and Vision, 1998.
- [39] C.Faisstnauer, D.Schmalstieg, and W.Purgathofer, *Priority Scheduling for Networked Virtual Environments*, IEEE Computer Graphics and Applications **20**(6) (2000).
- [40] ———, *Scheduling for very large virtual environments*, Proc. of the IEEE Virtual Reality, 2000.
- [41] S. Cheshire, *Latency and the Quest for Interactivity*, <http://www.stuartcheshire.org> (1996).
- [42] E. Cronin, B. Filstrup, and A. Kurc, *A Distributed Multiplayer Game Server System*, <http://www.eecs.umich.edu/~bfilstru/quakefinal.pdf> (2001).
- [43] E. Cronin, B. Filstrup, A. R. Kurc, and S. Jamin, *An efficient synchronization mechanism for mirrored game architectures (extended version)*, Kluwer Multimedia Tools and Applications **23** (2004).
- [44] E. Cronin, S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, *Constrained mirror placement on the internet*, IEEE Journal on Selected Areas in Communications (2002).
- [45] P. S. Davis and J. L. Ray, *Branch and bound algorithms for the capacitated facilities location problem*, Naval Res. Log. Quart. **16** (1969), 331–344.

BIBLIOGRAPHY

- [46] J. Dias, R. Galli, A. Almeida, C. Belo, and J. Rebordao, *mworld: A multiuser 3d virtual environment*, IEEE Computer Graphics **17(2)** (1997).
- [47] C. Diot and L. Gautier, *A Distributed Architecture for Multiplayer Interactive Applications on the Internet*, IEEE Networks Magazine **13(4)** (1999), 6–15.
- [48] Z. Duan, Z. Zhang, and Y. T. Hou, *Service Overlay Networks: SLAs, QoS and Bandwidth Provisioning*, Proc. of the 10th IEEE International Conference on Network Protocols (ICNP), 2002.
- [49] H. Dysart and N. D. Georganas, *Newclust: An algorithm for the topological design of two-level, multidrop teleprocessing networks*, IEEE Trans. on Communications **26** (1978), 55–62.
- [50] M. Faloutsos, P. Faloutsos, and C. Faloutsos, *On Power-Law Relationships of the Internet Topology*, ACM SIGCOMM, 1999.
- [51] H. Feltl and R. Raidl, *An improved hybrid genetic algorithm for the generalized assignment problem*, Proc. of ACM Symposium on Applied Computing, 2004.
- [52] W. Chi Feng and W. Chang Feng, *On the geographic distribution of online game servers and players*, Proc. of NetGames, 2003.
- [53] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang., *IDMaps: A Global Internet Host Distance Estimation Service*, IEEE/ACM Transactions on Networking **9(5)** (2001).

BIBLIOGRAPHY

- [54] M. L. Fisher G. Cornuelos and G. L. Nemhauser, *Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms*, Management Sciences **23** (1977), 789–810.
- [55] C. Greenhalgh and S. Benford, *Boundaries, Awareness and Interaction in Collaborative Virtual Environments*, Proc. of the 6th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE) (1997).
- [56] K. Gummadi, S. Saroiu, and S. Gribble, *King: Estimating Latency between Arbitrary Internet End Hosts*, Proc. of the ACM SIGCOMM Internet Measurement Workshop, 2002.
- [57] O. Heckmann, M. Piringier, J. Schmitt, and R. Steinmetz, *Generating Realistic ISP-Level Network Topologies*, IEEE Communication Letters (2003).
- [58] T. Henderson and S. Bhatti, *Networked games: a QoS-sensitive application for QoS-insensitive users?*, Proc. of the ACM SIGCOMM, 2003.
- [59] R. Hitchens, *Java NIO*, O'Reilly, 2002.
- [60] D. S. Hochbaum, *Approximation algorithms for np-hard problems*, PWS Publishing Company, Boston, MA, 1997.
- [61] M. Hori, T. Iseri, K. Fujikawa, S. Shimojo, and H. Miyahara, *Scalability Issues of Dynamic Space Management for Multiple-Server Networked Virtual Environments*, Proc. of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 2001.

BIBLIOGRAPHY

- [62] J. Huang, Y. Du, and C. Wang, *Design of the server cluster to support avatar migration*, Proc. of IEEE Virtual Reality, 2003.
- [63] id Software, <http://www.idsoftware.com>.
- [64] IDC, *Asia/Pacific Online Gaming Subscription Revenue in 2004*, http://www.idc.com/getdoc.jsp?containerId=pr2005_08_04_110417.
- [65] B. Patrovsky J. Mulligan, *Developing online games: An insider's guide*, New Riders Games, 2003.
- [66] J. Kim, J. Choi, D. Chang, T. Kwon, Y. Choi, and E. Yuk, *Traffic Characteristics of a Massively Multi-player Online Role Playing Game*, Proc. of NetGames, 2005.
- [67] A. Kirmse and C. Kirmse, *Security in Online Games*, Game Developer 4(4) (1997), 20–28.
- [68] ———, *Security in Online Games*, Game Developer 4(4) (1997), 20–28.
- [69] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, *Peer-to-Peer Support for Massively Multiplayer Games*, Proc. of IEEE INFOCOM, 2004.
- [70] A. K. Kuehn and M. J. Hamburger, *A heuristic program for locating warehouses*, Management Sc. 9 (1963), 643–666.
- [71] F. Kuhl, R. Weatherly, and J. Dahmann, *Creating computer simulation systems: an introduction to the high level architecture*, Prentice Hall, Upper Saddle River, NJ, 1999.

BIBLIOGRAPHY

- [72] B. S. Lee, W. Cai, S. J. Turner, and L. Chen, *Adaptive Dead Reckoning Algorithms for Distributed Interactive Simulation*, International Journal of Simulation: Systems, Science & Technology **1(1)** (2000), 21–34.
- [73] K. W. Lee, B. J. Ko, and S. Calo, *Adaptive Server Selection for Large Scale Interactive Online Games*, Computer Networks **49** (2005), 84–102.
- [74] J. Lui and M. Chan, *An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems*, IEEE Transaction on Parallel and Distributed Systems **13(3)** (2002).
- [75] M. Macedonia, M. Zyda, D. Pratt, and P. Barham, *Exploiting Reality with Multicast Groups*, IEEE Computer Graphics and Applications (1995).
- [76] M. Mauve, *Consistency in Replicated Continuous Interactive Media*, Proc. of the 2000 ACM Conference on Computer Supported Cooperative Work, 2001, pp. 181–190.
- [77] M. Mauve, S. Fischer, and J. Widmer, *A generic proxy system for networked computer games*, Proc. of NetGames, 2002.
- [78] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, *Local-lag and timewarp: Providing consistency for replicated continuous applications.*, IEEE Transactions on Multimedia **6(1)** (2004), 47–57.
- [79] D. Miller and J. Thorpe, *Simnet: The advent of simulator networking*, Proc. of the IEEE **83(8)** (1995).

BIBLIOGRAPHY

- [80] D. Min, E. Choi, D. Lee, and B. Park, *A Load Balancing Algorithm for a Distributed Multimedia Game Server Architecture*, Proc. of the IEEE International Conference on Multimedia Computing and Systems, 1999.
- [81] K. Moore, J. Cox, and S. Green, *SONARA Network Proximity Service (Internet Draft)*, <http://www.netlib.org/utk/projects/sonar> (1996).
- [82] K. L. Morse and J. Steinman, *Data Distribution Management in the HLA: Multidimensional Regions and Physically Correct Filtering*, Proc. of the Spring Simulation Interoperability Workshop, 1997.
- [83] NCSoft, *Lineage*, <http://www.lineage.com>.
- [84] Beatrice Ng, A. Si, R. Lau, and F. Li, *A Multi-Server Architecture for Distributed Virtual Walkthrough*, Proc. of the ACM Symposium on Virtual Reality Software and Technology (VRST), 2002.
- [85] T. S. Eugene Ng and H. Zhang, *Predicting Internet Network Distance with Coordinates-Based Approaches*, Proc. of the IEEE INFOCOM, 2002.
- [86] C. D. Nguyen, F. Safaei, and P. Boustead, *Comparison of distributed server architectures in providing immersive audio communication to massively multi-player online games*, Proc. of Australian Telecommunication Networks and Applications Conference (ATNAC), 2004.
- [87] T. V. Nguyen, F. Safaei, P. Boustead, and C. T. Chou, *Provisioning overlay distribution networks*, Computer Networks **49** (2005), 103–118.

BIBLIOGRAPHY

- [88] T. Nitta, K. Fujita, and S. Cono, *An application of distributed virtual environment to foreign language*, IEEE Education Society (2000).
- [89] K. Obraczka and F. Silva, *Network Latency Metrics for Server Proximity*, In Proc. of the IEEE Globecom 2000, 2000.
- [90] R. Govindan P. Radoslavov and D. Estrin, *Topology-informed internet replica placement*, Proc. of Web Caching and Content Distribution Workshop, 2001.
- [91] L. Pantel and L. Wolf, *On the Impact of Delay on Real-Time Multiplayer Games*, Proc. of Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2002, pp. 23–29.
- [92] J. Pellegrino and C. Dovrolis, *Bandwidth requirement and state consistency in three multiplayer game architectures*, Proc. of the ACM NetGames, 2003.
- [93] L. Qiu, V. Padmanabhan, and G. Voelker, *On the placement of web server replicas*, Proc. of IEEE INFOCOM, 2001.
- [94] R. Ramjee, *Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks*, Proc. of IEEE INFOCOM, 1994, pp. 680–768.
- [95] C. Reynolds, *Flocks, Herds, and Schools: A Distributed Behavioral Model*, ACM Computer Graphics **21(4)** (1987).
- [96] H.E. Romeijn and D. Romero Morales, *A class of greedy algorithms for the generalized assignment problem*, Discrete Applied Mathematics **103** (2000), 209–235.
- [97] S. Rooney, D. Bauer, and R. Deydier, *A federated peer-to-peer network game architecture*, IEEE Communication Magazine (2004).

BIBLIOGRAPHY

- [98] F. Safaei, P. Boustead, C. D. Nguyen, J. Brun, and M. Dowlatshahi, *Latency driven distribution: infrastructure needs for participatory entertainment applications*, IEEE Communication Magazine on Entertainment Everywhere: System and Networking Issues in Emerging Network-Centric Entertainment Systems **43**(5) (2005).
- [99] S. Singhal and M. Zyda, *Networked virtual environments: design and implementation*, Addison-Wesley, Reading, MA, 1999.
- [100] J. Smed, T. Kaukoranta, and H. Hakonen, *A Review on Networking and Multiplayer Computer Games*, Turku Centre for Computer Science, Technical Report (2002).
- [101] J. Smed, T. Kaukoranta, and H. Hakonen, *Aspects of Networking in Multiplayer Computer Games*, Proc. of the International Conference on Application and Development of Computer Games in the 21st Century, 2001, pp. 74–81.
- [102] SOE, *Planetside*, <http://planetside.station.sony.com/overview.jsp> (2003).
- [103] M. Szymaniak, G. Pierre, and M. van Steen, *Latency-Driven Replica Placement*, Proceedings of the International Symposium on Applications and the Internet (SAINT), 2005.
- [104] D. N. B. Ta and S. Zhou, *A Dynamic Load Sharing Algorithm for Massively Multi-Player Online Games*, Proc. of the 11th IEEE International Conference on Networks, 2003.

BIBLIOGRAPHY

- [105] G. Tan, L. Xu, F. Moradi, and S. Taylor, *An Agent-based DDM for High Level Architecture*, Proc. of the 15th Workshop on Parallel and Distributed Simulation, 2001.
- [106] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin, *The Impact of Routing Policy on Internet Paths*, Proc. of IEEE INFOCOM, 2001.
- [107] J. Volge and M. Mauve, *Consistency Control for Distributed Interactive Media*, Proc. of the ninth ACM International Conference on Multimedia, 2001, pp. 221–230.
- [108] B. M. Waxman, *Routing of Multipoint Connections*, IEEE Journal on Selected Areas in Communications **9** (1988), 1617–1622.
- [109] J. Yan and H.J. Choi, *Security Issues in Online Games*, The Electronic Library: International Journal for the Application of Technology in Information Environments **10(2)** (2002).
- [110] S. Zhou, W. Cai, B. S. Lee, and S. Turner, *Consistency in Distributed Interactive Applications*, European Simulation Interoperability Workshop 2001, 2000.
- [111] S. Zhou, W. Cai, B. S. Lee, and S. J. Turner, *Time-space consistency in large-scale distributed virtual environments*, ACM Transactions on Modeling and Computer Simulation **14(1)** (2004), 31–47.

BIBLIOGRAPHY

- [112] L. Zou, M. Ammar, and C. Diot, *An Evaluation of Grouping Techniques for State Dissemination in Networked Multi-User Games*, Proc. of the 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (2001).

Appendix A

Author's publications

During the period working on this thesis, I have published and submitted several refereed papers in international conferences and journals. These papers are listed as follows.

A.1 Journal papers

1. Duong N. B. Ta and S. Zhou, "A Network-centric Approach to Enhancing the Interactivity for Large-Scale Distributed Virtual Environments", *Elsevier Computer Communications Journal*, 2006 (**accepted, in press**).
2. Duong N. B. Ta and S. Zhou, "Client Assignment Problem in Distributed Virtual Environments", *International Journal of Computers and Applications*, 2005 (**accepted**).
3. Duong N. B. Ta, S. Zhou and H. Shen, "Greedy Algorithms for Client Assignment in Large-Scale Distributed Virtual Environments", *SCS Transactions on Simulation*, 2006. (**invited paper, under review**).
4. Duong N. B. Ta and S. Zhou, "A Two-phase Approach to Interactivity Enhancement for Large-Scale Distributed Virtual Environments", *Elsevier Computer Networks*, 2006 (**under review**).

APPENDIX A. AUTHOR'S PUBLICATIONS

A.2 Conference papers

1. Duong N. B. Ta and S. Zhou, "Server Placement for Enhancing the Interactivity of Large-Scale Distributed Virtual Environments", *5th International Conference on CYBERWORLDS*, EPFL, Switzerland, 2006 (**accepted, invited for The Visual Computer Journal, Springer**).
2. Duong N. B. Ta and S. Zhou, "Efficient Client-to-Server Assignments in Distributed Virtual Environments", *20th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Greece, 2006.
3. Duong N. B. Ta, S. Zhou, and H. Shen, "Greedy Algorithms for Client Assignment in Large-Scale Distributed Virtual Environments", *20th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS)*, Singapore, 2006 (**Best Paper Award nomination, invited for the SCS Transactions on Simulation**).
4. Duong N. B. Ta and S. Zhou, "Client Assignment Algorithms for Enhancing Interactivity in Distributed Virtual Environments", *4th IEEE International Conference on Computer Science: Research, Innovation and Vision for the Future (RIVF)*, Vietnam, 2006.
5. Duong N. B. Ta and S. Zhou, "Massively Multiplayer Online Games: Challenges and Our Approach", *1st International Workshop on Game Research and Development (CyberGames)*, Singapore, 2005.
6. Duong N. B. Ta and S. Zhou, "Client Allocation for Enhancing Interactivity in Distributed Virtual Environments", *5th International Conference on Computational Science and its Applications (ICCSA)*, Lecture Notes in Computer Science (LNCS) 3480, Springer-Verlag, Singapore, 2005.

APPENDIX A. AUTHOR'S PUBLICATIONS

7. Duong N. B. Ta, S. Zhou and H. Shen, "A Generic HLA-based Communication Architecture for Distributed Virtual Environments", *23rd International Conference on Parallel and Distributed Computing and Networks (PDCN)*, Acta Press, Austria, 2005.
8. Duong N. B. Ta and S. Zhou, "Effect of Inter-Server Communication in an HLA-based Distributed Virtual Environments", *1st IEEE International Conference on Distributed Frameworks for Multimedia Applications (DFMA)*, IEEE Computer Society Press, France, 2005.
9. Duong N. B. Ta and S. Zhou, "A Dynamic Load Sharing Algorithm for Massively Multiplayer Online Games", *11th IEEE International Conference on Networks (ICON)*, Australia, 2003.