A TOOL FOR NETWORK SIMULATION OF MASSIVELY MULTIPLAYER ONLINE GAMES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SELÇUK BOZCAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2008

Approval of the thesis:

A TOOL FOR NETWORK SIMULATION OF MASSIVELY MULTIPLAYER ONLINE GAMES

submitted by **SELÇUK BOZCAN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Volkan Atalay _____________________
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Veysi İşler _____________________
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Asst. Prof. Dr. Erol Şahin _____________________
Computer Engineering Dept., METU

Assoc. Prof. Dr. Veysi İşler _____________________
Computer Engineering Dept., METU

Asst. Prof. Dr. Kayhan İmre _____________________
Computer Engineering Dept., HACETTEPE UNI.

Dr. Atilla Özgit _____________________
Computer Engineering Dept., METU

Asst. Prof. Dr. Pınar Şenkul
Computer Engineering Dept., METU _____________________

Date: _____________________

ABSTRACT

A TOOL FOR NETWORK SIMULATION OF
MASSIVELY MULTIPLAYER ONLINE GAMES

BOZCAN, Selçuk
M.Sc., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Veysi İŞLER

September 2008, 63 pages

Massively multiplayer online games (MMOGs) have become highly popular in the last decade and now attract millions of users from all over the world to play in an evolving virtual world concurrently over the Internet. The high popularity of MMOGs created a rapidly growing market and this highly dynamic market has forced the game developers to step up competitively. However, MMOG development is a challenging and expensive process. In this study, we have developed a network simulation tool which can be used to model and simulate typical MMOGs that have client-server architectures. The main objective is to provide a simulation environment to MMOG developers that could be used to test, analyze and verify various aspects of the MMOG network architecture. We have also implemented a graphical user interface which allows constructing the simulation model visually. We have demonstrated the use of simulation tool by experimental simulations.

Keywords: Network Simulation, MMOG, MMORPG

ÖZ

DEVASA OYUNCULU ÇEVRİMİÇİ OYUNLAR İÇİN
BİR AĞ BENZETİMİ ARACI

BOZCAN, Selçuk
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Danışmanı: Doç. Dr. Veysi İŞLER

Eylül 2008, 63 sayfa

Devasa oyunculu çevrim içi oyunlar (MMOG) son on yıl içerisinde son derece popüler oldular. Günümüzde dünyanın dört bir yanından milyonlarca kişi, internet üzerinden eşzamanlı olarak oynanan ve sürekli evrimleşen sanal bir dünyaya sahip olan bu oyunlar tarafından çekilmektedirler. MMOG'lerin bu yüksek popülaritesi dünya genelinde oldukça hızlı bir büyüme gösteren bir market oluşturdu. Bu son derece dinamik market, oyun geliştirici firmaları rekabetçi bir tutum sergilemeye zorladı. Fakat MMOG geliştirme süreci zorlu ve pahalı bir süreçtir. Bu çalışmada biz, tipik sunucu-istemci mimarisine sahip MMOG ağlarını modellemek ve benzetimini yapmak amacıyla kullanılabilecek bir ağ benzetimi aracı geliştirdik. Asıl amacımız MMOG geliştiricilerinin oyun geliştirme sürecinde MMOG ağlarının farklı yönlerini test etmek, analiz etmek ve doğrulamak için kullanabileceği bir ağ benzetim ortamı sunmaktır. Bu çalışma sırasında benzetim modellerinin grafiksel olarak tasarlanabileceği görsel bir araç da geliştirilmiştir. Çalışmamız dâhilinde geliştirdiğimiz ağ benzetim ortamının kullanımını gerçekleştirdiğimiz çeşitli deneysel ağ benzetimleriyle örnekledik.

Anahtar Kelimeler: Ağ benzetimi, MMOG, MMORPG

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my supervisor Assoc. Prof. Dr. Veysi İşler for his continual support, guidance and encouragements throughout this thesis work.

I would also like to thank my colleague Mrs. Burcu Özbek for her support and understanding.

Lastly, I would like to express my deepest thanks to my parents and my dearest sister. Their unconditional support, understanding, encouragements and sacrifice made this work possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MMOG	Massively Multiplayer Online Game
MMORPG	Massively Multiplayer Online Role Playing Game
GTNetS	Georgia Tech Network Simulator
LAN	Local Area Network
MMOPW	Massively Multiplayer Online Persistent World
MUD	Multi User Dungeon
PLATO	Programmed Logic for Automated Teaching Operations
MMOFPS	Massively Multiplayer Online First Person Shooter
MMORTS	Massively Multiplayer Online Real Time Strategy
AOI	Area of Interest
NS-2	Network Simulator-2
OPNet	Optimized Network Engineering Tools

CHAPTER 1

INTRODUCTION

1.1 Background and Purpose

MMOGs (Massively Multiplayer Online Games) or sometimes referred as MMORPGs (Massively Multiplayer Online Role Playing Games) are types of multiplayer computer games which are played by a massive amount of concurrent players online over the internet. With the internet boom in the mid to late 90s, MMOGs have rapidly gained in popularity [1], [2]. Currently the worldwide online gaming market is dominated by the MMOGs and moreover, in five years time the market revenue of the MMOGs is expected to be doubled according to the DFC Intelligence online gaming forecast report [3]. This rapidly growing and highly dynamic market signifies the importance of MMOGs and continuously attracts the game developers from all over the world. However, MMOG development is not an easy task and it is significantly more difficult and it costs more than other multiplayer games [4]. There are many challenges exists with regard to development, service and security.

One of the most important and challenging part of the MMOG development process is the design of the network architecture of the game. MMOGs are multiplayer only games. Except few characters, nearly all characters in the game are controlled by online human players. At any time, several thousand of players can be online in a typical MMOG and this number can be as high as several hundred of thousands in some very popular MMOGs. Designing a network architecture that will utilize several real-time online servers in an efficient and reliable way, creating a smooth game play for massive amount of concurrent players is by no means an easy task.

Efficient and reliable network performance is crucial for most MMOGs. MMOG players expect constantly available game servers with minimum latencies [5]. If an MMOG fails to satisfy these requirements, most likely players will move to other MMOGs which promise better service quality and game play. If the development expenses of an MMOG are taken into account, it is clear that, a significant downfall in the subscriber count of the game could cause serious money loss for the vendor of the game.

In this study, we have developed a network simulation tool for MMOGs which can be used to model and simulate typical MMOG networks. Network simulation is used for various aspects of research and development of networks, including protocol analysis and verification, evaluating the network architectures and testing network scenarios [6]. Especially for complex networks, the use of network simulation tools for analysis, testing, verification and for other similar purposes has become a necessity.

We believe that using a network simulation tool can substantially help the development process of an MMOG. For example, the game network protocol can be analyzed and verified in the early stages of the development process, long before the deployment of the game. Early testing of the network protocol will hopefully improve the quality and decrease the bug count. It is clear that fixing a bug in the late stages of the development will cost much more. The cost will be even higher after the deployment.

The network simulation tool developed in this study is based on the general purpose network simulation environment GTNetS [7]. We aimed to develop an easy to use network simulation environment which is capable of modeling and simulating typical client-server based MMOG networks. Our motivation is to provide a network simulation environment to the MMOG developers, which can be used to test, analyze and verify various aspects of a typical MMOG network. We believe that, using a network simulation tool can decrease the development expenses of the MMOGs and improve the quality of the games.

1.2 Document Organization

This thesis work organized in 6 chapters. Chapter 2 discusses the MMOGs in detail. Important characteristics of MMOGs, background and history, typical MMOG architectures as well as MMOG development challenges are discussed in Chapter 2. Chapter 3 is about network simulation. General information about network simulation and discrete event network simulation is discussed in Chapter 3. Also this chapter includes a brief survey of widely used network simulators. Our rationale for choosing the GTNetS network simulator and related works that we have been motivated are discussed in Chapter 3. Chapter 4 describes the network simulation tool developed in this thesis work. The architecture and implementation details of the tool are given in Chapter 4. Chapter 5 includes experiments that we have conducted with the network simulation tool that is developed. Experiments and results are discussed in this chapter. Chapter 6 concludes this thesis work and states the future work.

CHAPTER 2

MASSIVELY MULTIPLAYER ONLINE GAMES

2.1 Background & Characteristics

A Massively Multiplayer Online Game (MMOG), as the name suggests, is a computer game which is played simultaneously by a massive amount of players online over the Internet. The most important characteristic of MMOGs, that distinguishes them from other multiplayer computer games, is the amount of simultaneous players. Multiplayer computer games which are played over LANs generally have between 10 and 40 simultaneous players. In case of MMOGs, the number of concurrent players online can be as high as several hundreds of thousands. For example *World of Warcraft* [8] which is one of the most popular MMOGs [1], [2] has achieved 1 million concurrent players in China at 11 April 2008 [9].

MMOGs or sometimes incorrectly referred as MMORPGs (Massively Multiplayer Online Role Playing Games), have become so popular in the last decade. Especially in Asia, MMOGs have reached phenomenal levels of success and everyday they are getting more popular in Europe and the U.S. [1], [2]. Millions of people are spending many hours while playing MMOGs. Figure 1 shows the total active MMOG subscriptions from all over the world. The tremendous growth of the MMOGs after late 90s can be seen clearly from the figure. Also in Figure 3, subscription counts of the most popular MMOGs (MMOGs that have over 300000 subscribers) are shown. These two figures should reveal the rapid growth in the MMOG industry. More data concerning about the MMOG growth as well as other MMOG market information can be found at MMOG Data [1] and MMOG Chart [2].

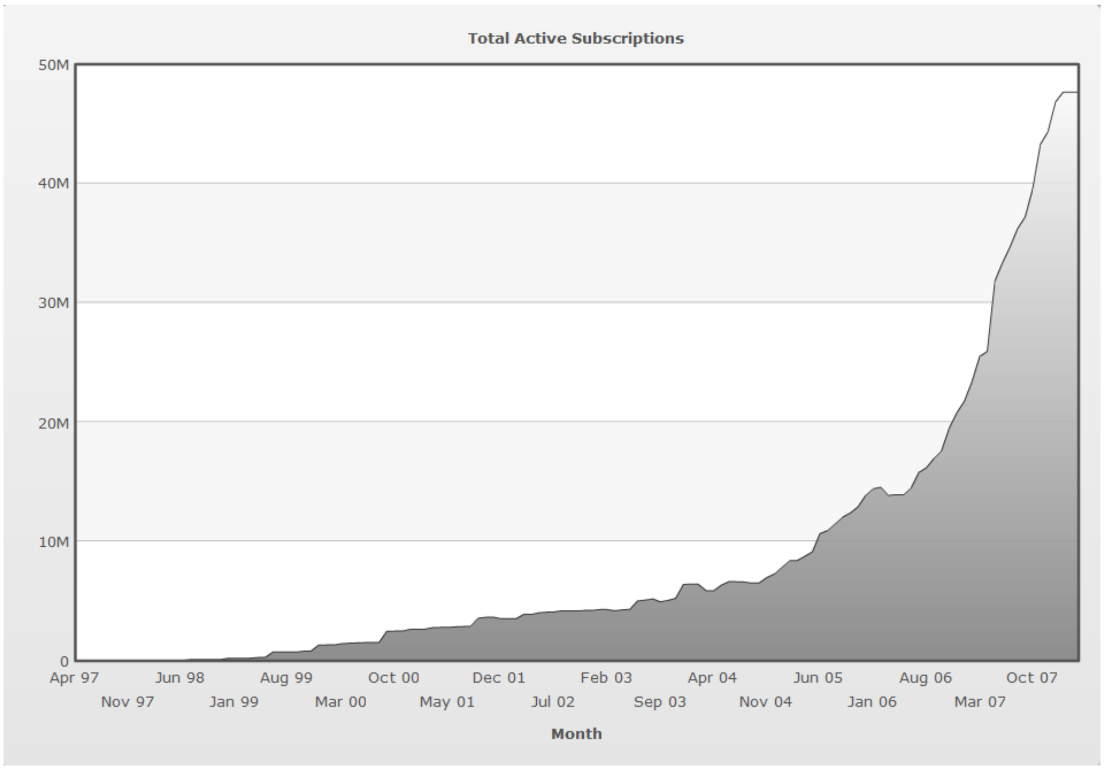


Figure 1 Total Active MMOG Subscriptions [1]

The virtual environments presented in the MMOGs are usually huge in size due to the massive amount of concurrent players. Moreover, only a little of the game characters, which are called NPCs (Non-Player Character), are played by the computer. The majority of the characters in the virtual environment are played by the humans. This important characteristic of the MMOG virtual environments leads to high interaction between the players. Such high interaction among the players creates a strong social networking environment which is not available in any other kind of computer game. Figure 2 shows a crowded market place from an MMORPG called *Metin 2* [10]. Especially in MMORPGs, hundreds of players can be online at any time in the towns of the game's virtual world.

Figure 2 Screenshot from Metin2

It is very likely that, the high popularity of MMOGs is mainly because of the high interaction levels between the players offered by the MMOG virtual environments. Ducheneaut et al. [11] have suggested that most of the activities that a player can perform in a typical MMOG are already present in the single player games. The main attraction of MMOGs arises from the shared game play experience, the collaborative nature of most activities and being socialized into a community of gamers and acquiring reputation within it. High interaction between the players increases the variability in the game and forms a non-linear gameplay. MMOGs usually don't have an ending in the sense of other computer games, which means that a player can't finish the game. This property of MMOGs makes it possible that the players can play the game for months or even years without getting bored.

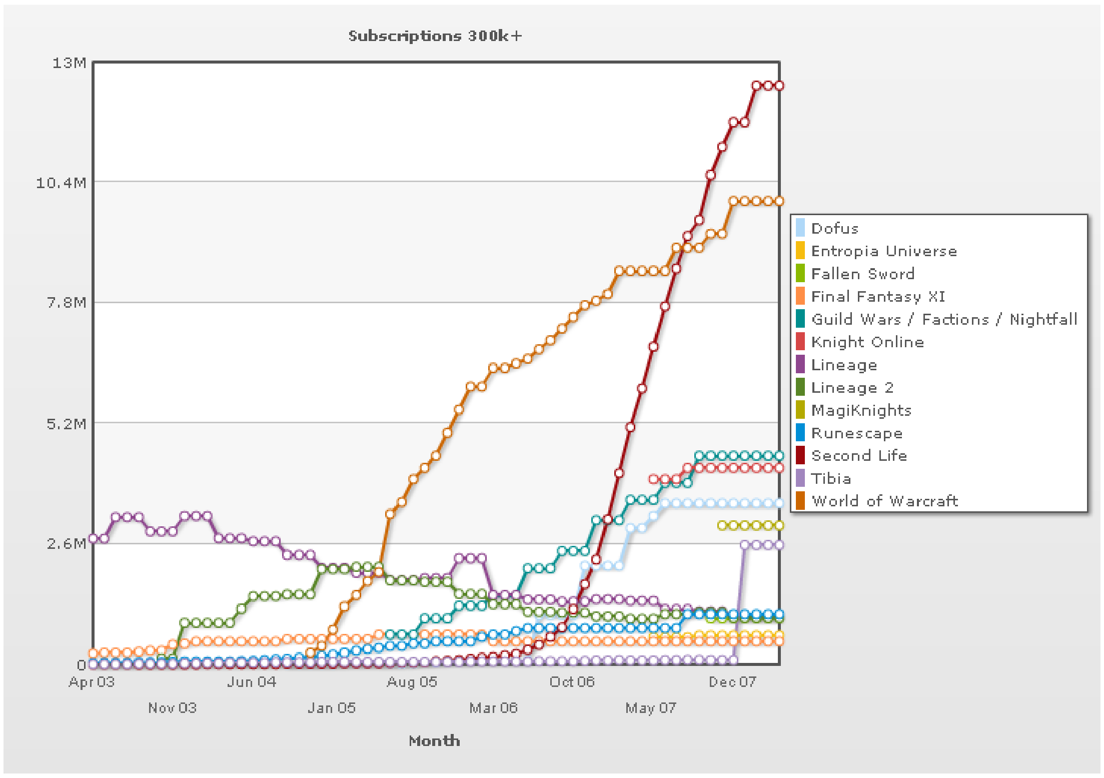


Figure 3 MMOG Subscriptions 300K+ [1]

In addition to high level of interaction, one of the most important characteristics of MMOG virtual environments is the persistency. Unlike other computer games, the virtual environments presented in MMOGs are persistent. In other words, they continue to exist and evolve even if no players are connected to it. The persistency of the virtual environments presented in MMOGs is so characteristic, that sometimes MMOGs are referred as MMOPWs (Massively Multiplayer Online Persistent Worlds) [12]. This specific feature of MMOGs makes them more immersive than their counterparts.

Table 1 summarizes the different non-technical key characteristics of MMOGs and other multiplayer games.

Table 1 Comparison of MMOGs and Multiplayer Games (Adopted from [13])

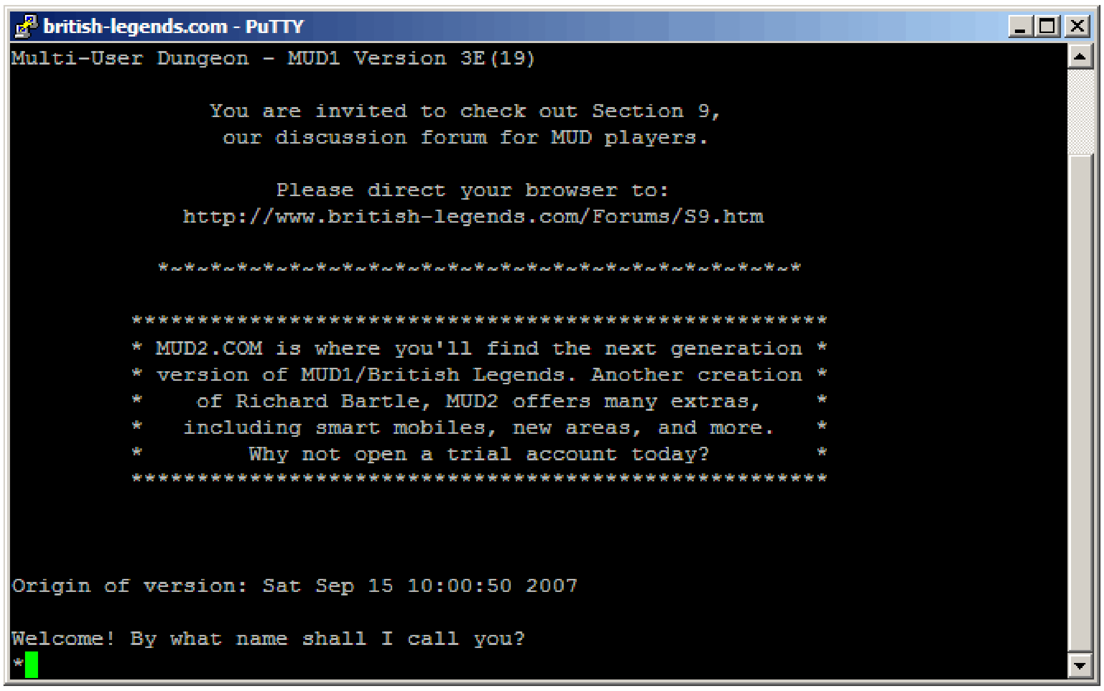
Characteristic	MMOGs	Multiplayer Games
Network	Internet only	Internet or LAN
Number of Players	Usually tens of thousands concurrent players, up to several hundreds of thousands	Usually in the order of tens, up to few hundred
Virtual Environment Size	Large	Small
Virtual Environment Type	Persistent	Transient / Session-based
Price Model	Subscription based, annual or monthly fees.	One time paid game costs

2.2 History of MMOGs

Unlike other multiplayer computer games, MMOGs are strictly Internet based. However long before the widespread availability of the Internet, there were games played on private LANs, which feature some of the important characteristics of MMOGs.

In 1973, *Maze War* (a.k.a *The Maze Game*) was the first computer game to introduce the concept of multiplayer [14]. The game featured a simple virtual environment which players can join and interact with each other. The game was originally implemented to enable networking over serial ports. But in 1974, it is updated to run over the ARPANET [15], which is the predecessor of the modern Internet. *Maze War* can be considered as the ancestor of all multiplayer games and for that reason it is a very important milestone in the history of MMOGs.

Another important milestone achieved in the history of MMOGs is the introduction of MUDs (Multi User Dungeons) [16] in 1978 [17], [18]. MUDs are generally accepted as the ancestors of modern MMORPGs [17], which happen to be a genre of MMOGs. MUDs were solely text based multiplayer computer games which are played over the TELNET protocol. The players interacted with the virtual environment presented in the game by typing simple commands like *walk, attack* and etc. In 1978, Roy Traubshaw and Richard Bartle created the first known MUD, which is named MUD1 [19], at Essex University on a DEC PDP-10. The MUD1 ran on the PLATO (Programmed Logic for Automated Teaching Operations) system, which is an educational network running on mainframes with graphical terminals. MUD1 attracted a lot of attention and eventually became very popular among the undergraduate students in a very short time. In Figure 4 a screen from MUD1 is given. Although compared to modern MMOGs, MUDs were very simple but yet, they featured some of the most important characteristics of the modern MMOGs, like large number of simultaneous players and persistent virtual worlds [16].



```
Multi-User Dungeon - MUD1 Version 3E(19)

               You are invited to check out Section 9,
                our discussion forum for MUD players.

                    Please direct your browser to:
             http://www.british-legends.com/Forums/S9.htm

           *~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*

         *****************************************************
         * MUD2.COM is where you'll find the next generation *
         * version of MUD1/British Legends. Another creation *
         *    of Richard Bartle, MUD2 offers many extras,    *
         *   including smart mobiles, new areas, and more.   *
         *        Why not open a trial account today?        *
         *****************************************************


Origin of version: Sat Sep 15 10:00:50 2007

Welcome! By what name shall I call you?
*
```

Figure 4 MUD1 Logon Screen [19]

The next big step in the history of MMOGs is the release of *Meridian 59* [20] in 1996. *Meridian 59* is the first commercial, truly Internet based, 3D massively multiplayer online game [21]. Actually, the term “massively multiplayer” is first used for the *Meridian 59.* The game was developed by a software company called Archetype Interactive which is later bought by 3DO. It cost $9.95 monthly to play the game. Even though *Meridian 59* was an inspiring game which brought many new aspects to MMOG world, unfortunately the success of it didn’t last very long. The popularity of the game was overrun by the release of one of the most successful MMOGs ever, *Ultima Online* [22]*,* only a year after.

After late 90s the popularity of MMOGs increased drastically (Figure 1) with the impact of MMOGs like *Ultima Online, Lineage* [23] and *EverQuest* [24]. Those games were became so successful and attracted millions of people. The outstanding success of early MMOGs, encouraged game developing companies to invest in MMOGs. As a result, as the years past more promising MMOGs emerged. For example, *World of Warcraft*, which now holds more than 20 percent of the market share [1], has over 10 millions of subscribers from all over the world [2].

MMOGs have now become the most popular type of online games and generate more than half of the online gaming market revenue. According to the DFC Intelligence [3], the worldwide market is going from $2.2 billion in 2006 to $5.9 billion in 2012. This rapidly growing MMOG market underlines the importance of online gaming sector and will attract more companies to invest in MMOGs in next several years.

2.3 Genres of MMOGs

MMOGs are often incorrectly referred as MMORPGs. In fact, MMORPGs are a genre of MMOGs. This misusage of the terms is mainly because of two reasons. The first one is that, the very first and successful MMOGs like MUDs, *Meridian 59, Ultima Online* and so forth were MMORPGs. Other types of MMOGs are developed later or didn’t achieve the same success. Second reason is more than 70 percent of modern MMOGs is of type MMORPG [1]. So that MMORPG genre has become the front face of MMOGs. However, there are actually several genres of MMOGs, some of which are listed below.

- **MMORPG:** MMO Role Playing Games are the most popular type of MMOGs. Examples are *World of Warcraft, Lineage* and *Guild Wars* [25].

- **MMOFPS:** MMO First Person Shooter. Examples are *World War II Online* [26]*, The Agency* [27].

- **MMORTS:** MMO Real Time Strategy. Examples are *Mankind* [28] and *Shattered Galaxy* [29].

- **MMO Social Game:** *Second Life* [30] developed by Linden Labs can be considered as a MMO Social Game since the game mainly focuses on creating a social environment.

- **Others:** There are also other types of MMOGs focused on areas like sports, dance and etc.

2.4 Architecture of MMOGs

Nearly all of the existing commercial MMOGs have client-server architecture [31]. The reasons for that it is easier to maintain game consistency, persistency, security and administrative control in client-server architectures when compared to other possible architectures [32]. The MMOG client builds up the user interface (UI) of the game and runs on the player's computer. Usually, the MMOG client is responsible for maintaining the communication between the MMOG server and client and performing audio – visual rendering. On the other hand, the MMOG server is responsible for everything else related with the game such as, receiving client responses, evaluating client actions within the game logic and sending the results back to the clients.

Hundreds of thousands of people can be online at any time in an MMOG. To be able to deal with such massive number of concurrent clients, different approaches and different server configurations are developed and applied. Usually in MMOGs, the server responsibilities are distributed among a group of machine rather than using a single machine to handle all the work [31]. There are many possibilities for distributing the responsibilities of the servers. One of the possible configurations is shown in the Figure 5.

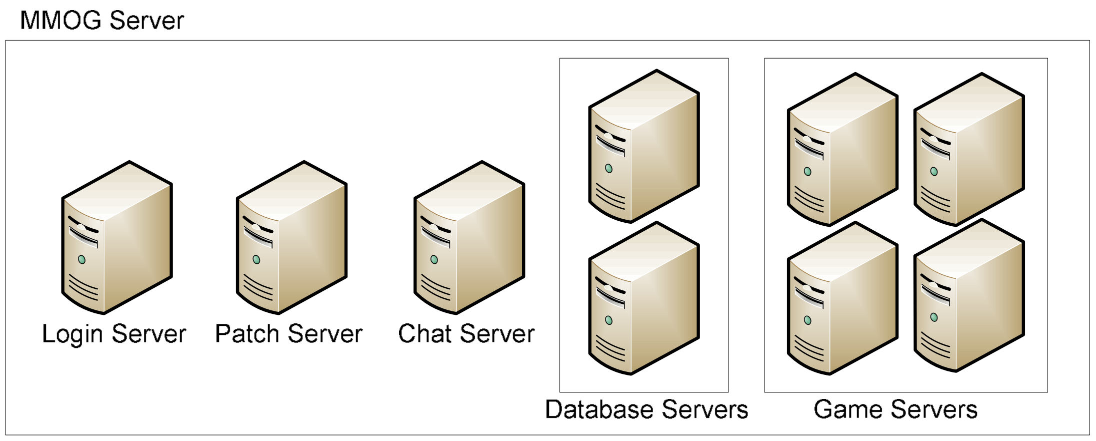


Figure 5 Example MMOG Server Model

In this sample server configuration, the server responsibilities split into server and server groups as follows: The login server is responsible for authentication of the clients. If the client succeeds the authentication process it can be forwarded to the patch server. The patch server is responsible for ensuring the up-to-dateness of the client software. If the client is not running the latest version, the patch server can send updates to the client. Chat server is responsible of handling the chat messages that the clients send. In this model, the game logic is handled by a group of servers. All game specific calculations are carried on the game servers. Different approaches are possible for distributing the work load among the game servers. Each server in the game server group can be statically assigned to a geographic location in the virtual environment or the game server can be dynamically configured to share the load according to different metrics [33]. Database servers store the game data, client avatar data and other configuration data related with the game. This server group, sometimes referred as MMOG server farm, acts as a single server in the MMOG network model. To be able to cope with massive amount of clients, sometimes MMOGs have multiple copies of this server farms which are distributed over different geographical locations of the world. For example *World of Warcraft* has North America, Europe and Asia servers [8]. These servers are totally independent from each other and each of them runs a copy of the game.

Centralized architectures are claimed to be unscalable by some researchers [31]. However, some of the existing implementations of MMOGs with client-server architecture have shown sufficient scalability. MMOGs like *World of Warcraft, EverQuest* and *Lineage* all have a centralized architecture, yet still support hundreds of thousands to be online concurrently. Nevertheless, it is still true that a decentralized P2P architecture can lower the operational costs by making use of computational power and bandwidth of the clients and improve the performance of the MMOG by reducing the latencies [34]. For that reason, there have been a number of proposals on P2P architectures for MMOGs [35], [36], [37], [38], [39]. There are also some hybrid architectures are proposed [34], [40]. However, these P2P and hybrid proposals are relevantly new and unfortunately are limited to the academic studies. To the best of our knowledge, there is no currently available P2P based commercial MMOG. However, *Outback Online* [41], which is an MMOG which is currently being developed by Yoick, may be the first commercial P2P based MMOG.

2.5 MMOG Development Challenges

MMOGs are significantly more difficult and expensive to develop than the traditional video games [4]. Development challenges arise from many points. First of all designing a virtual world at massive scales and acquiring seamlessness of this world, is itself a great challenge. A virtual world at this size together with the data produced by the hundreds of thousands of players brings a massive amount of data to deal with. Managing this data effectively on a very large and heterogeneous network is another challenge. Also an MMOG should maintain a smooth gameplay no matter how many players are online. It should be clear why developing an MMOG is extremely difficult when all this challenges are summed up. In fact, even big companies can fail to create a successful MMOG. For example, *Earth & Beyond* [42] which was a science fiction MMOG developed by Electronic Arts & Westwood Studios in five years time, released in September 2002 and shut down in September 2004 [4].

The development costs of a typical MMOG are higher than the typical computer games. However, operational and maintenance costs can be even higher than the development costs. The cost of maintaining a huge bandwidth for the operation of hundreds of thousands of simultaneous player connections makes up the largest share in operational and maintenance costs [4].

Because of the high development costs of MMOGs, it would be a better approach to measure the game performance on early steps of development period. The performance of the important components of an MMOG, like the network architecture, network protocol and interest management algorithm, should be measured at early stages of the development period in order to prevent late changes in that components, which will definitely increase the development expenses.

CHAPTER 3

NETWORK SIMULATION

Network simulation itself is an important part of this study, where we have extended a network simulation environment to model and simulate MMOG network architecture. In this chapter, an introduction of network simulation is given and discrete event network simulation is described in detail. This chapter also contains a survey of some of the most popular and commonly used network simulators. Finally we present the rationale for choosing GTNetS (Georgia Tech Network Simulator) as our simulation environment.

3.1 Introduction

Network architecture design is critical for multiplayer game development process. Especially in MMOGs, the proper design of the network architecture design is even more critical since, the success of an MMOG is heavily dependent on efficient and reliable network operation. However, because of the massiveness of the MMOGs, the network architectures could be very complex. Designing a properly operating game network over the Internet, which is a highly complex and heterogeneous environment, is a challenge that MMOG developers face.

In general, MMOG users expect robust game servers with little or none downtimes [5]. Server outages will cost real money and affect the market share of the publisher company, as the players move to games that promise a more reliable game play [43]. To give an example, recent outages in the *World of Warcraft* network have cost the Blizzard Entertainment an estimated $26198 per hour [44].

In addition to robustness of game servers, an MMOG network also should have low latency values [45]. In a computer network, latency means the amount of time it takes a message to travel from source node to the destination node. Smed, Kaukoranta, and Hakonen [46] note that the variance in latency values is known as "jitter", and they suggest that in MMOGs an acceptable latency should range from 100 ms up to 1 second. Higher values of latencies are only acceptable as long as there is minimal jitter. If the MMOG users are constantly experiencing high latencies it is quite likely that they will become dissatisfied with the game, which might eventually result in abandonment of the game by its users.

As the complexity of the network increases, their design and analysis gets more challenging. The complexity of MMOG network architecture has made the use of simulation methods necessary for analyzing the game network performance. Various aspects of the game architecture can be analyzed, tested and verified by using network simulation method as listed below:

- *Game Network Protocol:* The performance and the suitability of the network protocol used by the game can be analyzed and verified by using network simulation methods. Representation of a player in the virtual environment is sometimes called as *avatar.* For example, a game client can send *MOVE* messages to the server while the *avatar* is moving. The impact of sending *MOVE* messages every second or every 2 seconds can be measured using network simulation.

- *Distributed Architecture for MMOG Servers:* Network simulation methods can be used to justify the game server architecture. For example, questions like "How is the performance of the system when a single server used?", "How is the performance of the system affected when a multiple server configuration instead of a single server is used?" or "If we increase the server number, how will be the latency values in the game network will be affected?" can be answered by using network simulation.

- *Performance of Area of Interest (AOI) Algorithm:* Various AOI (a.k.a Interest Management Algorithm) algorithms are used by MMOGs to decrease the overall game network traffic [47]. The performance of different AOI algorithms can be measured and improved by using network simulation. The affect of the algorithm parameters on the game network traffic can also be observed. Therefore, it is possible to determine an optimal algorithm with optimal parameters for the MMOG, by using network simulation.

- *Virtual Map Size:* MMOGs (especially MMORPGs), usually divide the virtual environment into maps to distribute the client load among servers. Network simulation methods can help to determine an optimal virtual map size that will cause minimum server latency values.

Simulation has become highly valuable and commonly used method in designing and improving complex network architectures. Carefully designed set of one or more simulation based examples can help an MMOG developer significantly. Using network simulation might decrease the development time, increase the product quality and decrease the overall development costs. Therefore, it is clear that, network simulation should be an indispensible method for MMOG developers.

3.2 Discrete Event Simulation

Nearly all currently available network simulators are based on discrete event simulation for simulating the network behaviors [6]. In discrete event simulation, the operation of the simulation system is represented by the state changes that are caused by discrete events sorted with a chronological order. In a mathematical representation, the operation of the simulation system can be expressed as the following sequence [48]:

$$< s_0, (e_0, t_0), s_1, (e_1, t_1), s_2, (e_2, t_2) \ldots >$$

In the above sequence, s_i's are the system states, e_i's are discrete events and t_i's are nonnegative numbers representing event occurring times. The above sequence means that the simulation system is started with a state s_0. Then, at time t_0, event e_0 occurred and changed the system state to s_1. At time t_1, event e_1 occurred and changed the system state to s_2 and so on. Each event is assumed to happen instantly, meaning that takes zero time. It is important to note that, that $t_i \leq t_{i+1}$ for every i. Because, in a discrete event model two unrelated events can occur at the same time.

In discrete event simulation model, the simulation system is only interested in system states at discrete points in time and can ignore the system state changes at times in between those discrete points [6], [48]. For example, in a real network sending a data packet between two network devices that are connected with a point-to-point link can take several steps (state changes) including digital to analog, analog to digital conversions, error detection and correction, actual physical transmission of the data on the network medium. However, in a discrete event network simulation this scenario can be modeled with a simple approach [6]. The sending device starts to send data at some time T and it is received at the receiving node at time $T + \Delta T$. Here, ΔT represents the time needed for the sending node to transmit the entire data packet to the receiver node. It is clear that the state of the actual physical network varies during the time interval ΔT. However, the discrete event simulator can safely ignore these state changes. So, according to this approach, we only need to change the system state twice in order to model a simple data send between to directly linked nodes. The first state change occurs when the sending node completes the sending of data, and the second state change occurs the receiving node completely receives the data.

3.3 Network Simulators

In this section we will give a survey of some of the most popular network simulators used.

3.3.1 GTNetS

The Georgia Tech Network Simulator (GTNetS) [7] is a full featured network simulation environment that allows researchers in computer networks to study the behavior of moderate to large scale networks, under a variety of conditions [49]. It is an open source network simulator distributed under GPL. The source code of

GTNetS can be compiled under both Windows and UNIX based operating systems. According to its developers, the GTNetS design philosophy is to create a simulation environment much like actual networks are structured [49]. In GTNetS, there is a clear and distinct separation of protocol stack layers. Packets in GTNetS are made up of PDUs (Protocol Data Units) which are added and removed from the packet as it moves down and up in the protocol stack. As in the actual networks, network nodes in the simulator can have multiple network interfaces and each interface have an associated link and IP address. The network programming interface used by application objects in the simulator is quite similar to the ubiquitous POSIX standard.

GTNetS has support for a large number of applications and protocols. IEEE 802.3, IEEE 802.11 and Bluetooth protocols are implemented in GTNetS. Also it has support for numerous TCP and UDP based applications. Network models are written in C++. Object oriented approach used in GTNetS design makes it easy to extend the built in features.

GTNetS also supports distributed simulations using a federated simulation approach [49]. A single simulation can be distributed either on a network of loosely coupled workstations, a shared memory symmetric multi processor system, or a combination of both. GTNetS also uses dynamic NIx-Vector [50] routing method to reduce the memory footprint of the simulation. Parallelization ability and the NIx-Vector routing method used, grants GTNetS a high scalability. Because of this, large network simulations (networks consisting of several hundreds of thousands nodes) can be done with GTNetS.

A graphical user interface showing the representation of simulation topology also exists in GTNetS. The graphical user interface allows the user to enable or disable the visibility of selected nodes and links. Also, GTNetS can provide statistics regarding its own performance.

3.3.2 NS-2

The ns-2 (Network Simulator 2) [51] network simulator is one of the most well-known and widely used discrete event simulator [6] [52] [53]. It is developed and maintained by the Information Science Institute (ISI) at the University of Southern California and supported by DARPA and NSF. It is an open source project distributed under GPL.

Ns-2 has considerable support for TCP, routing and multicast protocols over both wired and wireless (local and satellite) networks [54] [53]. It includes detailed TCP models, large number of applications models and extensive logging and tracing support. Because of its popularity a large number of extensions for ns-2 can be found on the internet. Also the documentation of ns-2 is quite comprehensive. Ns-2 is developed mainly to run under UNIX based operating systems. However, on Windows OS it can be used through the Cygwin platform.

Ns-2 is based on two languages [55]. The core modules of ns-2 simulator are written in C++. The network configuration is defined using a scripting language OTcl [56] (an object oriented version of Tcl). There are two class hierarchies in ns-2. The compiled C++ class hierarchy and the interpreted OTcl class hierarchy. There is one to one correspondence between these two hierarchies. Ns-2 integrates C++ and OTcl implemented classes by constructing a linkage between them using TclCL [57]. The implementation architecture of ns-2 is shown in Figure 6.

Original ns-2 simulator can model network topologies up to 1000 network elements with reasonable performance. By using NIx-Vector routing method, ns-2 can model network topologies up to 16000 nodes [52]. Also a parallel distributed extension of ns-2 (pdns) [58] is developed by Riley, Fujimato and Ammar which can simulate networks with several hundreds of thousands of nodes.

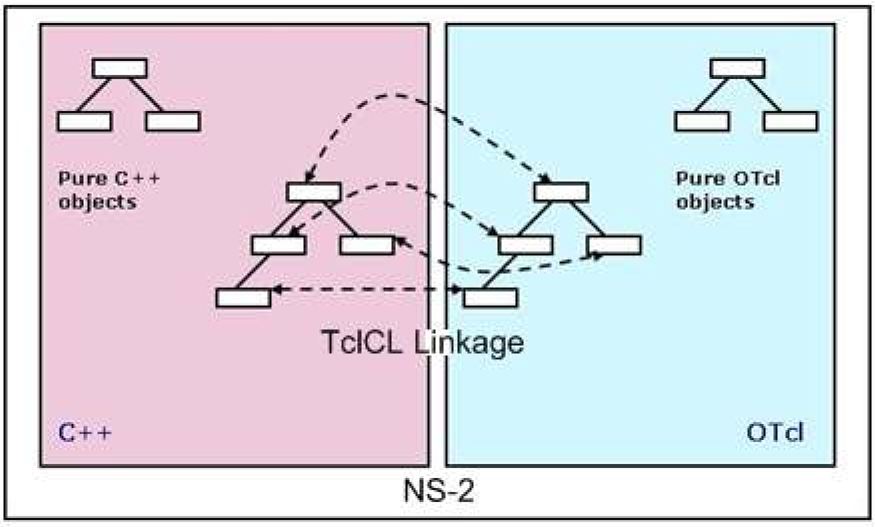


Figure 6 Implementation Architecture of ns-2 (Adopted from [51])

3.3.3 *OPNet*

OPNet (Optimized Network Engineering Tools) [59] is a commercial discrete event network simulation environment. In addition to discrete event simulation method, it also supports analytical and hybrid simulation methods. OPNet is the most widely used commercial network simulator [53]. OPNet Modeler features a comprehensive simulation environment which allows the design and analysis of networks, devices, protocols and applications. OPNet supports an extensive list of protocols and allows users to collect detailed statistics regarding the simulation. Additionally, it features a convenient user interface which allows users to design, debug and analyze the simulation topology graphically.

OPNet supports distributed simulations. It can execute and monitor several simulation scenarios in a concurrent manner.

3.4 Network Simulation Environment Selection

Developing a network simulation environment from scratch would be time consuming and is not in the scope of this study. Therefore, we have decided to select a simulation environment as a basis for our study and extend it as needed.

The simulation environment for this thesis study is selected after a comparison and evaluation of a group of commonly used simulators. It should be noted that each network simulator has its own features and none of them can be said to be better than the others. Hence, we tried to select the most appropriate network simulation environment which can meet our requirements. In our study, we needed a network simulation environment which is can sustain very large configurations of the simulated network. Also it should be easily extendible so that new network protocols and new network applications can easily be added to the simulation environment. Lastly, it should provide an easy to use programming interface and adequate documentation.

As we have surveyed the network simulators, GTNetS seemed to be a reasonable choice for the simulation environment which will be used in this study. First of all, the low memory footprint of GTNetS [52] was important for us since, it is possible to model larger networks with a low memory consuming network simulator.

Furthermore, the extendibility of the network simulation environment was also important for us. We believed that, extending GTNetS would be easy because of the open source license and the object-oriented design. Besides, GTNetS can be directly compiled and used on both Windows and UNIX based operating systems, which would eventually give us more freedom to select the study environment.

To give a better insight of GTNetS network simulation environment, we will explain the usage of GTNetS with a simple simulation example. While the simulation example (adopted from [6]) discussed here is simplistic, the steps described here are common to all network simulations in general.

1. <u>Create the Simulator Object:</u>

In GTNetS, the overall behavior of the simulation is controlled by a singleton object called Simulator.

```
Simulator s;   // Create the simulator object
```

2. <u>Create Network Nodes:</u>

Network nodes are represented with the Node class in GTNetS. In our example there will be two network node, a sender and a receiver.

```
Node* sender = new Node();       // Create sender node

Node* receiver = new Node();     // Create receiver node
```

3. <u>Create Links:</u>

In GTNetS network links are represented with the Link class. In our example will add a point-to-point link between nodes. In GTNetS, point-to-point links are represented with Linkp2p class which is inherited from Link class.

```
Linkp2p p2pLink(Rate("100Mb"), Time("2ms")); // Create link
sender->AddDuplexLink (receiver, p2pLink,
                       IPAddr("192.168.1.1"),
                       IPAddr("192.168.1.2"));
```

The first parameter of the Linkp2p constructor specifies the bandwidth of the link whereas the second parameter specifies the delay for the link. `AddDuplexLink` member function specifies the remote end point for the link. Optionally link specification and IP addresses for local and remote end point can also be specified.

4. <u>Add Applications:</u>

We will add an application that will send 1000000 bytes over the TCP protocol to the sender node. Also we will add a TCP server application to the receiver node which will listen incoming connections from a specified port.

```
TCPServer* rcv =
   (TCPServer*)receiver-> AddApplication(TCPServer());
rcv->BindAndListen(1234);       //Listen port 1234
rcv->Start(0);                  // Start at time 0

TCPSend* snd =
(TCPSend*) sender->AddApplication(TCPSend(receiver->GetIPAddr(),
         1234, Constant(1000000)));
snd->Start(0);
```

5. <u>Run the Simulation:</u>

After the simulation model is created, the simulation could be run. In GTNetS, the simulation is started by calling the Run member function of the simulator object.

```
s.Run(); // Run the simulation
```

6. <u>Print the Statistics:</u>

After the simulation run we can now gather the statistics about regarding our simulation. In GTNetS, numerous statistics are collected by default. In this example we will only print the throughput.

```
std::cout << "TCP throughput is"
         << ((TCP*)rcv->GetL4()))->GoodPut() << "bytes/sec";
```

The simulation scenario described above was given in GTNetS. But however, the basic steps described here are still applicable to other simulation environments.

3.5 Related Work

The popularity of MMOGs is increased drastically in the last decade. This high popularity of MMOGs attracted many researchers from different disciplines to study various aspects of MMOGs. In this section we will discuss some of the related work which are about MMOG network architecture and simulation of MMOGs.

Kuan ta Chen et.al analyzed a 1356-million-packet trace from a MMORPG called *ShenZou Online* in their study [60]. This study is important because, as far as we know it is the first formal analysis of MMORPG server traces. In their study, they have identified important characteristics of MMORPG traffic. They have listed the important properties of MMORPG traffic as; tiny packets, strong periodicity, temporal dependence of packet arrivals within the connections and aggregate traffic, irregularity, self similarity and heavy-tailed session duration. Also they have offered explanations for these characteristics of MMORPG traffic. They have stated that the periodicity is due to a common network game design pattern that tends to send out periodic state updates such as position changes. The temporal locality in game traffic is largely due to the game's design. The irregularity is due to the diversity and the huge size of the game's virtual environment. The self-similarity of the aggregate traffic is due to the heavy-tailed active/idle activities of individual players. Moreover, they have investigated the suitability of the TCP protocol for MMOGs and found that the TCP protocol produces a significant overhead due to the TCP/IP headers and acknowledgements. They have argued that use of TCP protocol for MMORPGs can be considered as overkill.

Another similar but important study about traffic analysis of MMOGs has been done by Philipp Svoboda et.al. [61]. In their study they have analyzed network traffic of the popular MMOG *World of Warcraft* [8]. Throughout their analysis they have extracted parameters that represent the characteristics of the traffic. They have used these parameters to implement a simulation model using the network simulator ns-2.

In their study YungWoo Jung et al. [62] have purposed a system that provides an automated beta test environment to efficiently test online games. They have called their system as *VENUS* (Virtual Environment Network User Simulator). In *VENUS* system they have implemented virtual clients which can communicate with the multiplayer game server. The virtual clients are implemented in a way that the game server can't distinguish them from actual clients. The virtual clients are controlled from a central station. The user can control the *VENUS* system from this central station. The central engineering station (CES) sends control commands to the VCs and the VCs send game messages to the game server. The game server is monitored by the *VENUS* system and the statistics regarding its performance are collected. Figure 7 shows the block diagram of the *VENUS* system.

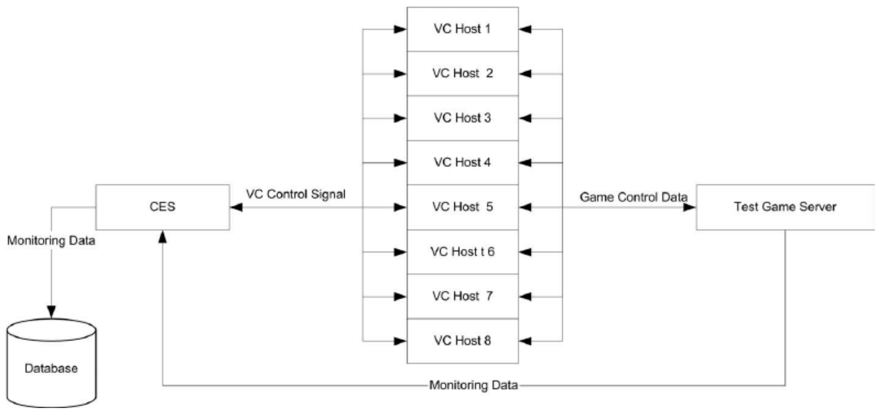


Figure 7 Block diagram of the VENUS system [62].

VENUS system tries to create an automated beta test environment to test multiplayer games. The virtual clients in the *VENUS* system connect to actual game server and try to mimic the behaviors of real clients according to the given commands. In our study however, we developed a network simulation environment which can be used to model and simulate the typical MMOG networks. *VENUS* system is developed to be used in beta testing phase of the development. In contrast, a network simulation tool can be used at very early stages of the development. For example, a network simulation tool can be used to evaluate the performance of a specific area of interest algorithm or the performance of a specific game network protocol, even before the implementation of the game server. We believe that, it would give better results to use both simulation based techniques and the automated test environments in the development of MMOGs.

CHAPTER 4

A TOOL FOR NETWORK SIMULATION OF MASSIVELY MULTIPLAYER ONLINE GAMES

4.1 Requirements

This study is mainly related to network simulation of Massively Multiplayer Online Games (MMOGs). More specifically, the aim of this study is to develop a network simulation environment which can be used to model and simulate typical MMOG networks that have client and server architecture.

The network simulation tool for the MMOGs should have good scalability and could be used to simulate large scale networks. Unlike typical multiplayer games, MMOGs have large number of concurrent players. To be able to simulate a typical MMOG network, the simulation environment should be capable of simulating networks with at least several thousand nodes.

The network simulation tool should enable its users to easily define custom game messages with variable sizes and frequencies. There is no standard for MMOG network protocol. Every MMOG uses its own custom defined message set for server and client communication. Therefore, allowing its users to define their own custom messages is a required feature.

MMOG network simulation tool should provide mechanisms for defining area of interest algorithms. Area of Interest (AOI) algorithms are utilized by the MMOG servers to decrease the network traffic on the game network. Since this algorithm significantly affects the network traffic, it is an important feature of network simulation tool for MMOGs.

The tool should support standard TCP protocol. Since nearly all of the commercially available MMOGs use TCP as transport protocol [61] [60], having TCP support is a required feature of network simulation tool for MMOGs.

The tool should enable its users to easily model and simulate a typical MMOG network. Modeling of large networks is a complex task and can be very time consuming. Therefore, a network simulation tool for MMOGs should provide functionalities to ease the modeling of the MMOG network.

4.2 Architecture Of The Tool

The MMOG network simulation tool developed in this study is based on the GTNetS [7] network simulation environment. We have extended and added new functionalities to the GTNetS to support network simulation of typical MMOG networks. Furthermore, we have implemented a modeler tool that allows visual modeling of MMOG networks. The visual modeling tool that we have developed generates a script file which represents the simulation model. Then, this script file is taken as input by our simulation environment and the corresponding simulation model is created. For this functionality, we have implemented an extendible parser module for the network simulation environment. After creating the simulation model in the simulation environment and running the simulation, the simulation environment reports the simulation results to the user. The overall architecture of the MMOG simulation tool is given in the Figure 8.

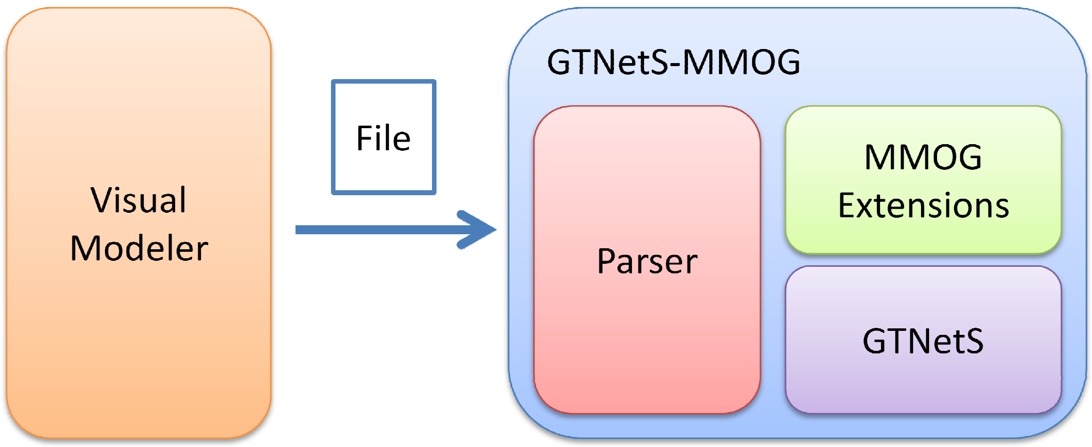


Figure 8 Architecture of MMOG Simulation Tool

We have called the network simulation environment as GTNetS-MMOG to emphasize that it is a network simulation environment based on GTNetS and has specific features to model and simulate MMOG networks. The Visual Modeler tool and the modules of the GTNetS-MMOG are described in detail in the following sections.

4.2.1 GTNetS and MMOG Extensions

GTNetS is a general purpose and extendible network simulation environment. Although, it has comprehensive support for various network protocols (i.e. TCP, UDP, HTTP) and application types (i.e. FTP, Web Browser, Internet Worm), it lacks some of the required features to model a typical MMOG network. However, GTNetS is open source and provides an extensible architecture. Therefore, we have extended GTNetS and added new message types, application types and other features to easily model and simulate an MMOG network.

4.2.1.1 Contributions to GTNetS Code

During the time of our study we made an extensive use of GTNetS and conducted various simulation experiments. Throughout our use of GTNetS we found several bugs in the GTNetS code and proposed solutions for those bugs. For simplicity, we will list here only two important bugs that we have found and proposed a fix.

- *TCP State Machine Bug:* During our experiments we realized that after a connection close event issued by a TCP application in GTNetS, the state of the TCP connection is never set to CLOSED. We found that, this is because of a small bug in the TCP state machine implementation of GTNetS. The state of the TCP connection should be set to CLOSED if a TIMEOUT event occurs while the TCP connection is in TIMED_WAIT state. We performed required corrections in the GTNetS code and reported this bug to the developers of GTNetS.

- *Data PDU Deletion Bug:* In our study we have implemented custom data messages in GTNetS. During the implementation of our custom message structures we discovered that a data PDU (Protocol Data Unit) with custom data never gets deleted in GTNetS even after it is processed and marked to be deleted. This bug causes huge memory leaks in the large scale network simulations and decreases the performance of the simulator significantly. We proposed a solution for this bug by adding deletion codes to the necessary points in the GTNetS code.

4.2.1.2 MMOG Extensions to the GTNetS

We added new applications and data messages to the GTNetS in order to model and simulate typical MMOG networks. The extensions that we have provided can be used to model and simulate a generic MMOG server and a generic MMOG client with custom game messages. The structure of the extensions provided is described in the Figure 9.

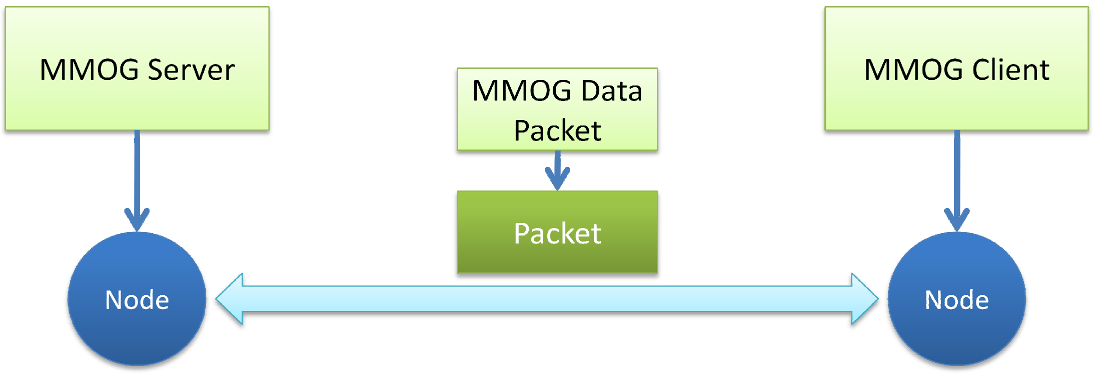


Figure 9 GTNetS MMOG Extensions

We have implemented two application objects in GTNetS, which are *MMOG Server* and *MMOG Client*. As their name suggests, *MMOG Server* application represents a typical MMOG server and *MMOG Client* application represents a typical MMOG client. Since nearly all of the existing commercial MMOG implementations use TCP as transport protocol [61] [60], we have implemented *MMOG Server* and *MMOG Client* applications to use TCP for communication. Additionally, we have also implemented a custom data packet which allows defining custom message structures that will be used in client-server communication. GTNetS MMOG extensions have been implemented using C++ programming language and the MS Visual Studio 2005 IDE. The detailed descriptions of the MMOG Server and Client as well as MMOG Data Packet are given in the following sub-sections.

4.2.1.2.1 MMOG Server

MMOG Server is a TCP application model which represents a typical MMOG server. It allows incoming connections from *MMOG Client* application connections from a configurable port. As the clients connect, *MMOG Server* gives a unique id to the clients and stores them in a structure for later references.

MMOG Server can be configured to operate with respect to an Area of Interest (AOI) algorithm. We implemented a simple Euclidean distance area of interest algorithm, however different types of interest management algorithms can easy be added as needed.

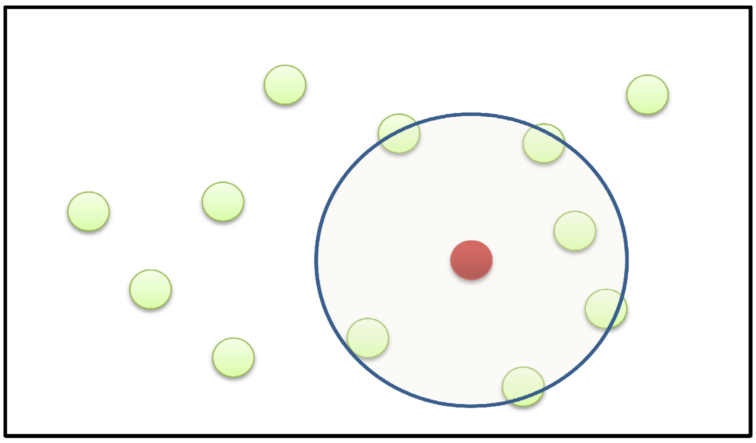


Figure 10 Euclidean Distance AOI

In Figure 10, Euclidean distance area of interest algorithm is shown. The small circles in the figure represent the players in the virtual environment. The area of interest is a circle around the player with a radius that covers the maximum distance a player can see. In the figure the area of interest of the red colored player is shown. If the distance between two players is smaller than the radius of the area of interest, then the updates of players are sent to each other. Otherwise, the player updates are filtered. Area of Interest algorithms increases the network performance by decreasing the amount of messages on the network.

In our tool, we have implemented Euclidean area of interest algorithm with configurable radius value. When a *MMOG Client* sends a message that is marked as “distribute to other clients with respect to AOI”, the *MMOG Server* creates copies of this message and send to the *MMOG Clients* within the area of interest of the sender client.

4.2.1.2.2 MMOG Client

MMOG Client is a TCP application model which represents a typical MMOG client. *MMOG Client* connects to a *MMOG Server* and sends game messages with respect to defined parameters. For area of interest algorithm, we have also implemented an extendable mobility model for the *MMOG Client* application. The mobility model that we have implemented simulates the movement of a player in the virtual environment by generating random waypoints in a defined virtual area. We have implemented our mobility model in such a way that, other models can be easily implemented and integrated through our interface. As the player moves in the virtual environment the *MMOG Client* sends messages to the *MMOG Server* reporting its position in the virtual environment. This position information is used by the *MMOG Server* in area of interest calculations. The position messages of the *MMOG Client* are not mutable. However, the size and the update frequency of the message can be adjusted similar to the custom game messages.

We have implemented a generic message sending feature in the *MMOG Client*. Size and the sending frequency of the custom messages can be adjusted. Rather than using only constant parameters to define the size and the frequency of the game messages, we have implemented a flexible structure that enables using of random distributions. We have implemented support for using following random distributions:

- Uniform: A constant probability distribution function, i.e. all the values in the distribution interval are equally probable [63]. Figure 11 shows the probability density function and cumulative distribution function of a continuous uniform distribution.

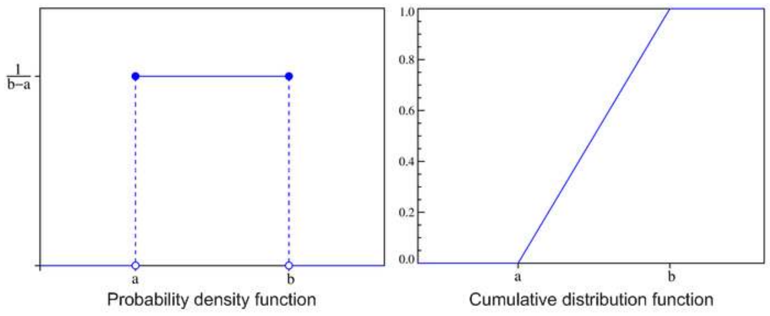


Figure 11 Uniform distribution (Adopted from [63])

- Exponential: A continuous probability distribution function whose density function is given by $f(x) = ae^{-ax}$, where $a > 0$ for $x > 0$, and $f(x) = 0$ for $x \leq 0$ [64]. Figure 12 shows the probability density function and cumulative distribution function of an exponential distribution.

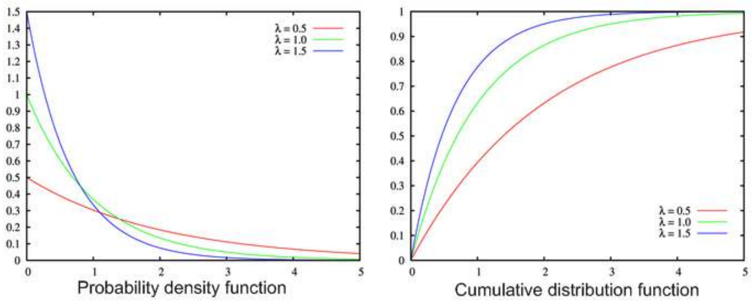


Figure 12 Exponential distribution (Adopted from [64])

- Pareto: For a random variable X with Pareto distribution, the probability that X is greater than some number x is given by $\Pr(X > x) = (\frac{x}{x_m})^{-k}$ for all $x \geq x_m$, where x_m is the (necessarily positive) minimum possible value of X, and k is a positive parameter [65]. Figure 13 shows the probability density function and cumulative distribution function of a Pareto distribution.

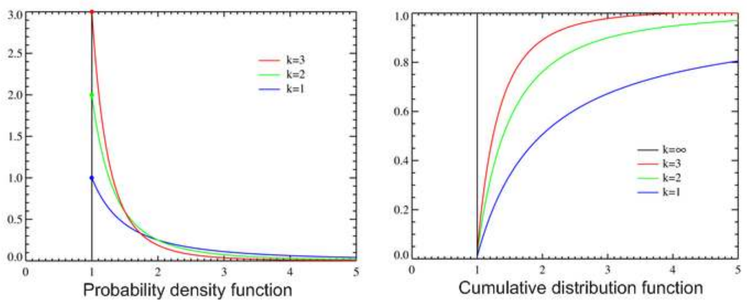


Figure 13 Pareto distribution (Adopted from [65])

- Weibull: Is a continuous probability distribution whose probability density function is given as $f(x;k;\lambda) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$ [66]. The probability density function and the cumulative distribution function of a Weibull distribution is given in Figure 14.

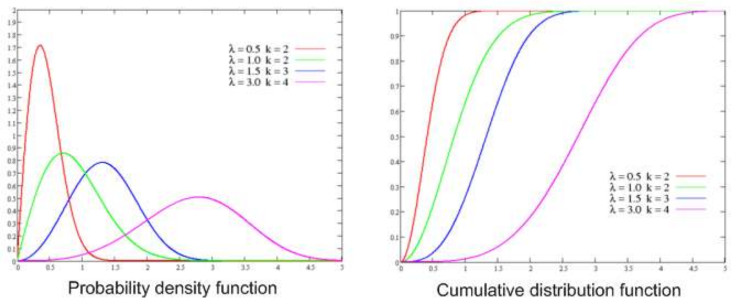


Figure 14 Weibull distribution (Adopted from [66])

Along with these random distributions, also constant values can be used to specify the size and update frequency of the custom messages.

4.2.1.2.3 MMOG Data Packet

To enable using of data messages that can include custom message structures we have implemented a *MMOG Data Packet* class which inherits from standard data packet in GTNetS. *MMOG Client* and *MMOG Server* communicate with using this message structure. The structure of the *MMOG Data Packet* is given in the Figure 15.

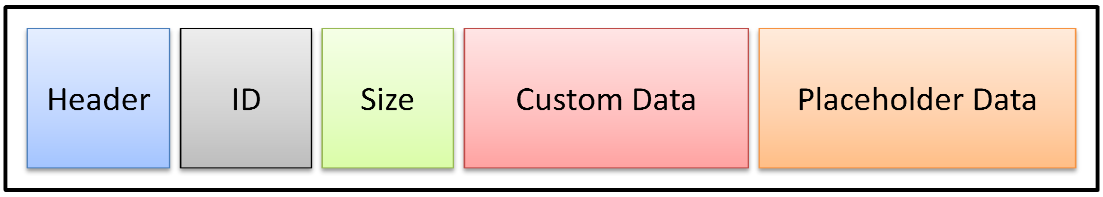


Figure 15 MMOG Data Packet Structure

Header of the message includes a specific signature which can be used to recognize a *MMOG Data Packet*. The ID field contains the unique id of the message. Size field contains size of the actual data in the message. Custom Data field contains the user defined custom data structure. The user can define larger size for the *MMOG Data Packet* than actually required for the user defined data. In that case, Placeholder Data is added to the end of the data packet.

4.2.2 Visual Modeler

In GTNetS, network simulations are written in C++ programming language. In small scale network simulations this method can be used with no problem at all. However, as the complexity and the size of the simulation model increases, the size of the C++ code required to generate the simulation model also increases and gets more complicated. Therefore, we have implemented a *Visual Modeler* tool for allowing the user to easily create and model typical MMOG networks without writing C++ code. *Visual Modeler* tool have been implemented with C# programming language and

Microsoft Visual Studio 2005 IDE. Screenshot of the main screen of *Visual Modeler* tool can be seen in the Figure 16.

With the *Visual Modeler* tool, the user can define the network topology by creating network nodes and links. The network nodes are represented with squares in the visual model. In *Visual Modeler* tool, a network node can be associated with either *MMOG Server* or *MMOG Client* applications. Alternatively, a network node can be created with no associated application at all. In that case, the created node will behave as a router in the MMOG network. Nodes with *MMOG Server* application associated are represented with red squares, whereas nodes with *MMOG Client* application are represented with blue squares. Additionally, router nodes are represented with black squares. The user can define the node IP and the application which will run on the node during the node creation process.

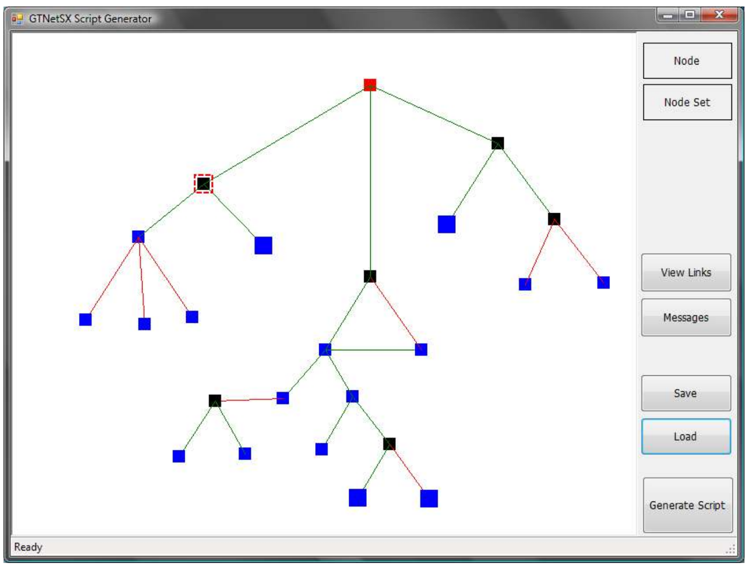


Figure 16 Screenshot of Visual Modeler Tool

Design of large scale networks can be time consuming and challenging even with using the *Visual Modeler* tool. For example, while designing a network with 1000 network nodes, the user should add each node and create links between them manually. Therefore, to ease this task we have added a feature which allows a user to define a node group with a single operation. Creation of a node group is similar to creation of node. While creating a node group, starting IP of the node group and the number of nodes in the group are defined. The IP of the nodes in the node group are assigned sequentially starting with the given IP value. Also, nodes in the node group can be associated with *MMOG Client* application. The node groups in the visual model are represented with larger squares than nodes have.

In *Visual Modeler* tool, the network links between the nodes can be created easily by simply selecting two nodes with mouse. The links are defined with two parameters: the bandwidth and the link delay. Additionally, a color can be assigned with the link, which makes the visual model more understandable. Currently, *Visual Modeler* tool only allows creation of full duplex links.

Creation of a link between two nodes is straightforward: a duplex link is created between two nodes (see Figure 17, (A)). In cases where one end of the link is a node group, a duplex link is created between the node and each node in the node group (see Figure 17, (B)). In cases where the both ends of the link is a node group, duplex links are created between each node in the first node group and each node in the second group (see Figure 17, (C)). There are no links between the nodes in a node group. Incoming and outgoing links of node and node groups are displayed in node properties screen (see Figure 18). Updating the links of a node or node group can be done from that screen.

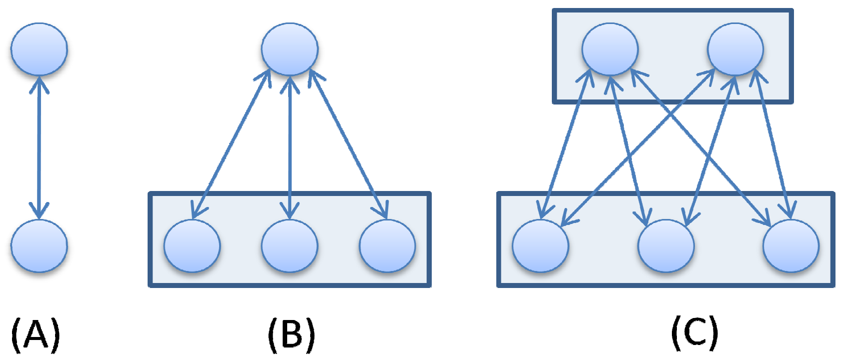
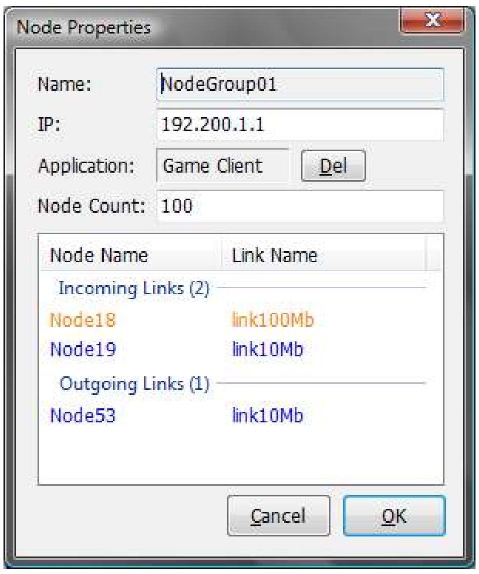


Figure 17 Visual Modeler Link Creation

Visual Modeler tool also allows defining custom game messages. A game message can be defined with two parameters, one for specifying the message size and the other for specifying the message sending frequency. For both parameters, a random distribution or a constant value can be selected. Selectable random distributions are uniform, exponential, Pareto and Weibull. In addition to that, parameters of the position message of the *MMOG Client* can also be adjusted.



Name:	NodeGroup01
IP:	192.200.1.1
Application:	Game Client
Node Count:	100

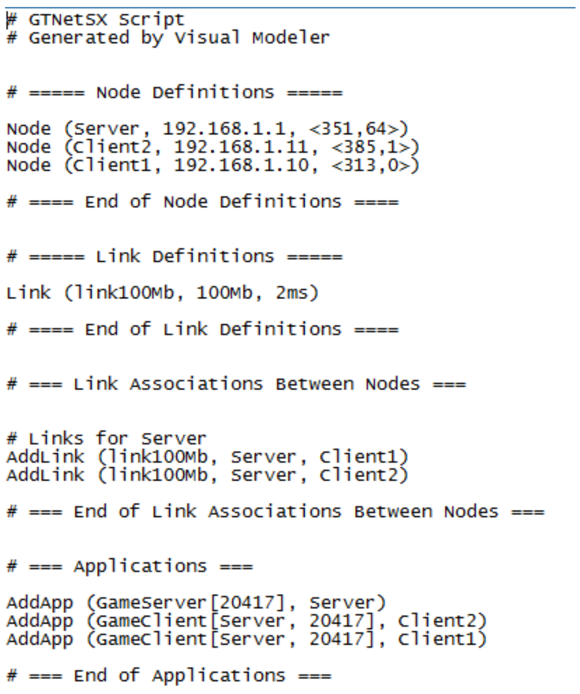
Node Name	Link Name
Incoming Links (2)	
Node18	link100Mb
Node19	link10Mb
Outgoing Links (1)	
Node53	link10Mb

Figure 18 Node Properties Screen

Saving and loading of simulation models are also supported in *Visual Modeler* tool. We have decided to use XML formatted files for storing the simulation models instead of using binary files. By using XML files, we aimed that saved simulation model files can be modified with using simple text editors.

Visual Modeler tool generates a script file which includes the definition of the simulation model. This script file can be parsed with the parser module of the *GTNetS-MMOG* and the simulation is run. A sample script generated by the *Visual Modeler* tool can be seen in the Figure 19. This script defines a simple network which have one three nodes. One of the nodes (Server) is associated with *MMOG Server* application and the two others are associated with *MMOG Client* applications. There is one link definition which has a bandwidth of 100Mb and a delay of 2ms. The clients are connected to the server with this link.

The structure of the script files will be discussed in detail in the following sections.



```
# GTNetSX Script
# Generated by Visual Modeler

# ===== Node Definitions =====

Node (Server, 192.168.1.1, <351,64>)
Node (Client2, 192.168.1.11, <385,1>)
Node (Client1, 192.168.1.10, <313,0>)

# ==== End of Node Definitions ====

# ===== Link Definitions =====

Link (link100Mb, 100Mb, 2ms)

# ==== End of Link Definitions ====

# === Link Associations Between Nodes ===

# Links for Server
AddLink (link100Mb, Server, Client1)
AddLink (link100Mb, Server, Client2)

# === End of Link Associations Between Nodes ===

# === Applications ===

AddApp (GameServer[20417], Server)
AddApp (GameClient[Server, 20417], Client2)
AddApp (GameClient[Server, 20417], Client1)

# === End of Applications ===
```

Figure 19 Sample Script Generated by Visual Modeler

4.2.3 The Parser

We have implemented a parser module for the *GTNetS-MMOG*. The parser module parses *GTNetS-MMOG* scripts and creates the simulation environment. A sample script file is given in Figure 19.

We have used the Spirit [67] parser framework which is a part of Boost C++ Libraries [68]. Spirit is an object oriented recursive descent parser framework implemented using template meta-programming techniques [67]. Spirit mimics the syntax of Extended Backus Normal Form (EBNF) completely in C++ using operator overloading and template programming. Spirit parsers are backtracking and top down, which are capable of parsing rather ambiguous grammars efficiently. For example, the following sample code shows a parser definition in Spirit which parses a comma separated list of real numbers.

```
real_p >> *(ch_p(',') >> real_p)
```

The keywords that are accepted by the parser module are given in Table 2.

Table 2 Parser Keywords

Keyword	**Explanation**	**Example Usage**
`Node (name, IP, <x,y>)`	Creates a node with given name and IP. X and Y optionally define the position of the node in the simulation window.	`Node (node1, 192.168.1.1, <3,4>)`
`Link (name, bandwidth, delay)`	Creates a link definition with given name, bandwidth and delay. This link definition later can be used to link the nodes.	`Link (link1, 100Mb, 2ms)`
`AddLink (linkname, node0, node1,…, nodeN)`	Creates duplex links between node0 and node1, node0 and node2 and so on. Link name specifies the link definition to be used.	`AddLink (link1, server, client1, client2)`

Keyword	**Explanation**	**Example Usage**
`AddLink2 (linkname, node0, node1,…, nodeN)`	This command is similar to `AddLink` except that it creates links between each node in the list to all other nodes.	`AddLink2 (link1, server, client1, client2)` (client1 and client2 are also linked)
`AddApp (app, node1, node2,…,nodeN)`	Associates application with the specified nodes. Possible values for app are given in Table 3.	`AddApp (GameServer[4873], server1)`
`ClientLocMsg (size, frequency)`	Defines the size and the frequency parameters of the *MMOG Client*'s position message. *size* and *frequency* are random distribution definitions. Possible values are given in Table 4.	`ClientLocMsg (Constant(32), Uniform(1,3))`
`AddCustomMsg (name, size, frequency)`	Defines a new custom message to be used by the *MMOG Client. size* and *frequency* are random distribution definitions. Possible values are given in Table 4.	`AddCustomMsg(chat, Uniform(100, 200), Uniform(10, 20))`

The application definitions that are accepted by the `AddApp` keyword are given in the Table 3.

Table 3 Application Definitions for AddApp Keyword

Keyword	**Explanation**	**Example Usage**
`GameServer[port]`	Creates a *MMOG Server* application which listens specified port number.	`GameServer[4897]`
`GameClient[server, port]`	Creates a *MMOG Client* application which connects to specified server node. *port* denotes the server's listen port number.	`GameClient[server1, 4897]`

The random distribution definitions that are accepted by the parser module are given in the Table 4.

Table 4 Random Distribution Definitions for Parser

Keyword	**Explanation**	**Example Usage**
`Constant (c)`	A constant value specified with c.	`Constant (5)`
`Uniform (min, max)`	A uniform random distribution with specified min and max values.	`Uniform (10, 20)`
`Exponential (mean, bound)`	An exponential random distribution with specified mean and bound values.	`Exponential (1.0, 5.0)`
`Pareto (mean, alpha, bound)`	A Pareto random distribution with specified mean, alpha (shape) and bound values.	`Pareto (1, 1.5, 10)`
`Weibull (mean, alpha, bound)`	A Weibull random distribution with specified mean, alpha and bound values	`Weibull (1, 1.5, 5)`

CHAPTER 5

EXPERIMENTS & RESULTS

In order to observe the performance and the usability of the *GTNetS-MMOG* network simulation tool and the *Visual Modeler* tool which are developed in this study, we have conducted network simulation experiments. Our experiments are focused on different aspects of the MMOG network architecture. This chapter describes the network simulation experiments that we have conducted and comments on the results.

5.1 Simulation Environment

The network simulations described in this chapter were carried on an x86 PC running Intel Pentium at 3.2GHz with 3GB of memory. The operating system of the simulation computer was Windows XP Professional with SP2 installed. *GTNetS-MMOG* and the *Visual Modeler* tools were compiled for Windows XP operating system with using Microsoft Visual Studio 2005 development environment. Throughout the network simulations, no tasks other than the critical system tasks were allowed to run.

5.2 AOI Algorithm Performance Evaluation

The effect of the area-of-interest (AOI) algorithms' performance on the MMOG network traffic is significant. In this experiment, we have used our *GTNetS-MMOG* network simulation tool to measure the effect of interest circle's radius in a Euclidean distance area-of-interest algorithm on the MMOG network traffic.

5.2.1 Simulation Model

In this experiment, we have modeled a MMOG network consisting of 6000 MMOG clients, single MMOG server and 18 routers. The network topology in this experiment is inspired by the *Campus Network* topology [6]. We have used the *Visual* Modeler tool to model the network. Figure 20 shows the network topology in the *Visual Modeler* tool.

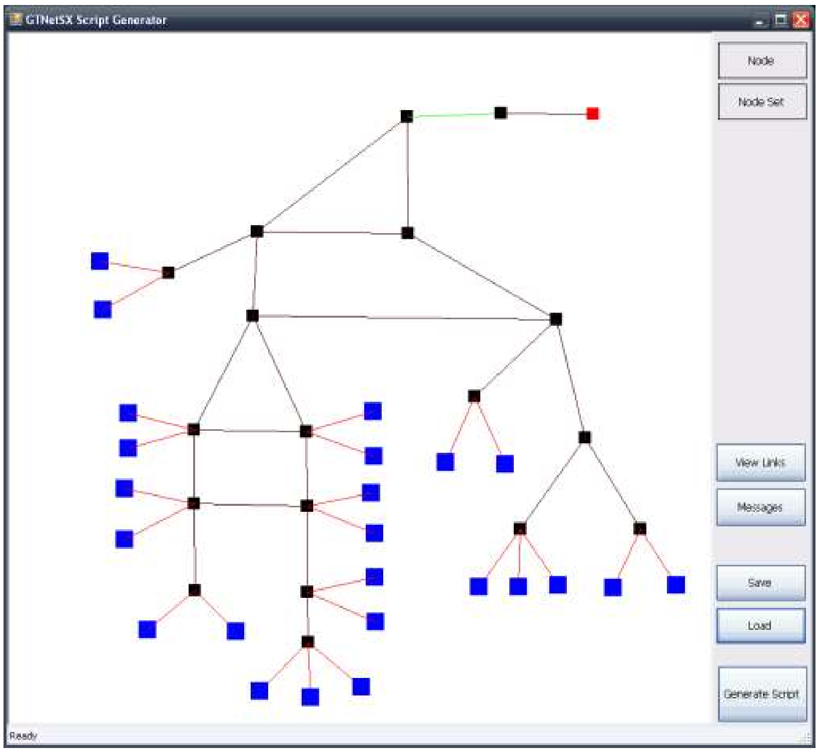


Figure 20 6000 Client MMOG Network Model

In the Figure 20, the blue squares represent the node groups. Each node group consists of 250 MMOG clients. The black squares represent the routers and the red square represents the MMOG server. The MMOG clients are connected to the routers with links that have 1Mbps of bandwidth and 2ms of delay (shown with red lines). Except from two routers in front of the MMOG server, all the routers are connected with links that have 16Mbps of bandwidth and 5ms of delay (shown with black lines). The two routers in front of the MMOG server are connected with a link that has 16Mbps of bandwidth and 1ms of delay (shown with green line). All the links in the network topology are in point-to-point fashion.

In the experiment, we have defined a virtual environment of size 300x300. All the MMOG clients are moving randomly within the virtual environment and sending their position updates to the MMOG server. The position messages of the MMOG clients are 50 bytes long. The MMOG clients send their position updates to the MMOG server every 1 to 3 seconds. We have modeled the sending of the position message with a uniform distribution that has a minimum as 1 second and maximum as 3 seconds. We run the network simulation 5 times for different values of the interest circle's radius. Each simulation is run for 30 minutes of simulation time.

5.2.2 Results

The network simulations took an average of 4.5 hours to complete on the simulation computer. The simulations, which the radius of the interest circle is smaller, took slightly less time than the simulations which the radius is larger. During the simulations, we have collected statistics about the MMOG server latency, the packets send by the server and clients and the packets received by the server. The simulation results are summarized in the Table 5.

Table 5 AOI Algorithm Simulation Results

Radius of Interest Circle	5	8	11	14	17
Best Server Latency	0.0191	0.0191	0.0191	0.0191	0.0191
Worst Server Latency	0.0401412	2.70992	2.7099	2.73434	2.70671
Average Server Latency	0.029232	0.0292434	0.0292292	0.0292245	0.0292242
Server RCV (Packets)	5384258	5385682	5385530	5385716	5386353
Server RCV (Bytes)	269212900	269304500	269286150	269292000	269323200
Server SND (Packets)	35703313	93382550	176558826	285258667	418639786
Server SND (Bytes)	1785165650	4669481950	8828267850	14263265150	20932433900
Client SND (Packets)	5384323	5386107	5385782	5385910	5386531
Client SND (Bytes)	269216150	269305350	269289100	269295500	269326550

From the simulation results, we observed that, as the radius of the interest circle increases linearly, the packet send by the MMOG server increases exponentially as expected. This relation between the radius of the interest circle and the packets send and received by the MMOG server is given in Figure 21.

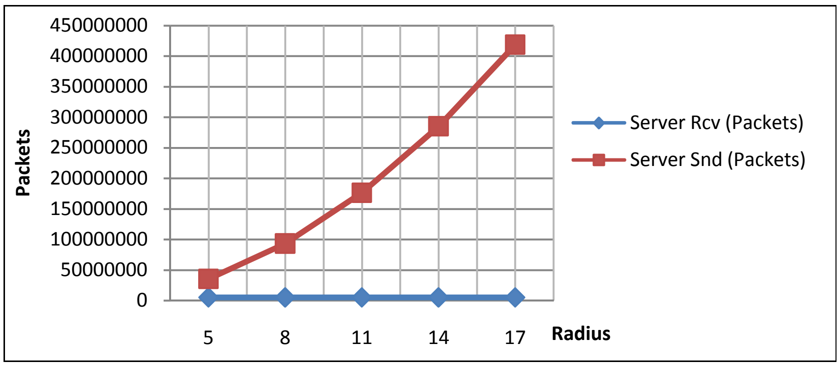


Figure 21 AOI Algorithm Simulation Server Statistics

The MMOG server and client bandwidth requirements can be calculated directly from these results. Table 6 shows the calculated average bandwidth requirements for MMOG server and MMOG client.

Table 6 AOI Algorithm Simulation: Server and Client Bandwidth Requirements

Interest Circle Radius	5	8	11	14	17
Server Bandwidth (B/s)	991758	2594157	4904593	7924036	11629130
Client Bandwidth (B/s)	24.92	24.93	24.93	24.93	24.93

The simulation results show that, an MMOG client would require an average of 24.92 B/s bandwidth for sending 50 bytes position update messages to the server every 1 to 3 seconds. On the other hand, MMOG server would require an 11629130 B/s bandwidth for only distributing the position updates of the clients when the radius of the interest circle is 17 units. Although the bandwidth requirement for a single client can be considered as tiny, the bandwidth requirement of the server reaches serious amounts as the radius of the interest circle increases.

5.3 MMOG Server Bandwidth Requirement Analysis

In typical MMOG networks, the bandwidth requirement of a single client is usually very low. However, MMOG servers require enormous bandwidths due to massive number of clients. In this experiment we have modeled number of simple MMOG networks with varying client counts and observed the bandwidth requirements of the MMOG server and the client.

5.3.1 Simulation Model

The MMOG network model that we have used in this experiment is similar to the one in the AOI algorithm performance evaluation experiment (see Figure 20). By changing the node counts in the node groups, we adjusted the MMOG client count on the network. We have modeled 7 MMOG networks with client sizes 960, 1920, 2880, 3840, 4800, 5760 and 6720 respectively.

For this simulation, in addition to the position update messages of the MMOG clients, we have defined two new messages. The first one is *chat* message, which can have a size between 100 bytes and 200 bytes. The *chat* message is sent by the MMOG clients in a period between 5 seconds to 10 seconds. We have used uniform distributions to model this behavior. The second message that we have introduced is the *action* message which can have a size between 50 bytes to 200 bytes. The *action* message is sent by the MMOG client every 5 to 15 seconds. The position messages are 50 bytes long and are sent to the MMOG server in every 1 to 3 seconds as in the previous experiment.

We have defined a virtual environment of size 300 x 300. Additionally, we have defined a Euclidean distance area-of-interest algorithm with 8 units of interest circle radius. Each simulation is run for 30 minutes of simulation time.

5.3.2 Results

During the simulations we have collected statistics about the messages sent and received by the MMOG server and clients. Table 7 summarizes the collected statistics. Note that the client send statistics are for average values for a single MMOG client.

Table 7 MMOG Server Bandwidth Requirement Analysis Results

Client Count	960	1920	2880	3840	4880	5760	6720
Server RCV (Packets)	1263027	2526697	3789908	5052191	6309364	7562362	8816587
Server RCV (Bytes)	98832631	197630902	296452097	395356756	494130891	593117911	691673342
Server SND (Packets)	3489314	13975444	31521597	55716605	86209742	125137882	171784391
Server SND (Bytes)	273126945	1093321952	2466443921	4360936250	6753706833	9816203742	13479137198
Client SND (Packets)	1316	1316	1316	1316	1294	1316	1316
Client SND (Bytes)	102952	102934	102936	102937	101221	102951	102928

Calculated bandwidth requirement values from these results are given in the Figure 22. From these results we have observed that as the client count increases linearly, the server bandwidth requirement increases exponentially. The exponential increase of the server bandwidth requirement, negatively affects the scalability of the overall MMOG network.

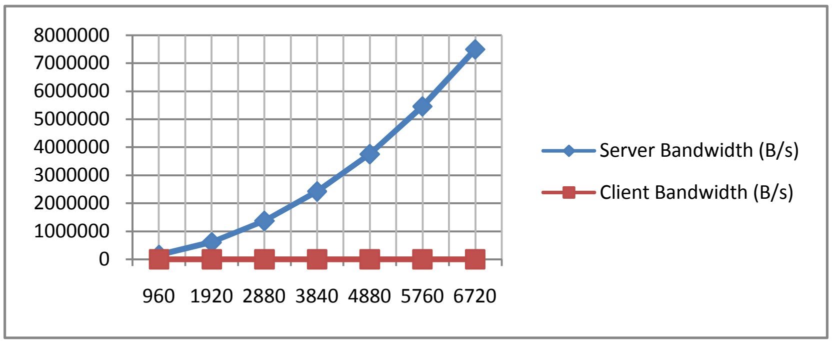


Figure 22 MMOG Server and Client Bandwidth Requirements

From the simulation results, it can be seen that the bandwidth requirements of the MMOG servers could reach to serious amounts. The maintenance cost of this much bandwidth could be very high for an MMOG developer company. Therefore, methods for decreasing the bandwidth requirement of the server should be applied. For example, many MMOG vendors distribute their servers geographically over the world to decrease the number of clients per server.

5.4 Virtual Environment Partitioning Analysis

In some MMOGs, in order to deal with massive amount of concurrent players, the virtual environment featured by the game is hosted by several servers. In this type of MMOGs, usually the virtual environment is geographically divided into regions and each region is associated with a server. As the clients move in the virtual environment, they send their updates to the server which is responsible of their region in the virtual environment.

In this experiment, in order to observe the network performance of the overall MMOG network with different virtual environment partitioning, we modeled four MMOG network simulations. In the first simulation model the MMOG virtual environment is hosted by a single server. In the other simulation models, the virtual environment is geographically divided into regions and each region is associated with a server, such as in the second simulation model the virtual environment is hosted by two servers, whereas in the third and fourth simulation model it is hosted by three and four servers respectively.

5.4.1 Simulation Model

For this experiment, we modeled four network simulation models with 2400 clients and 18 routers by slightly modifying the network simulation models used in the previous experiments. In the first simulation model there is a single server, whereas in the other simulation models there are two, three and four servers respectively. The link properties and the link connections are the same with the simulation model described in AOI performance evaluation experiment. Figure 23 shows the MMOG network model with four MMOG servers in *Visual Modeler* tool.

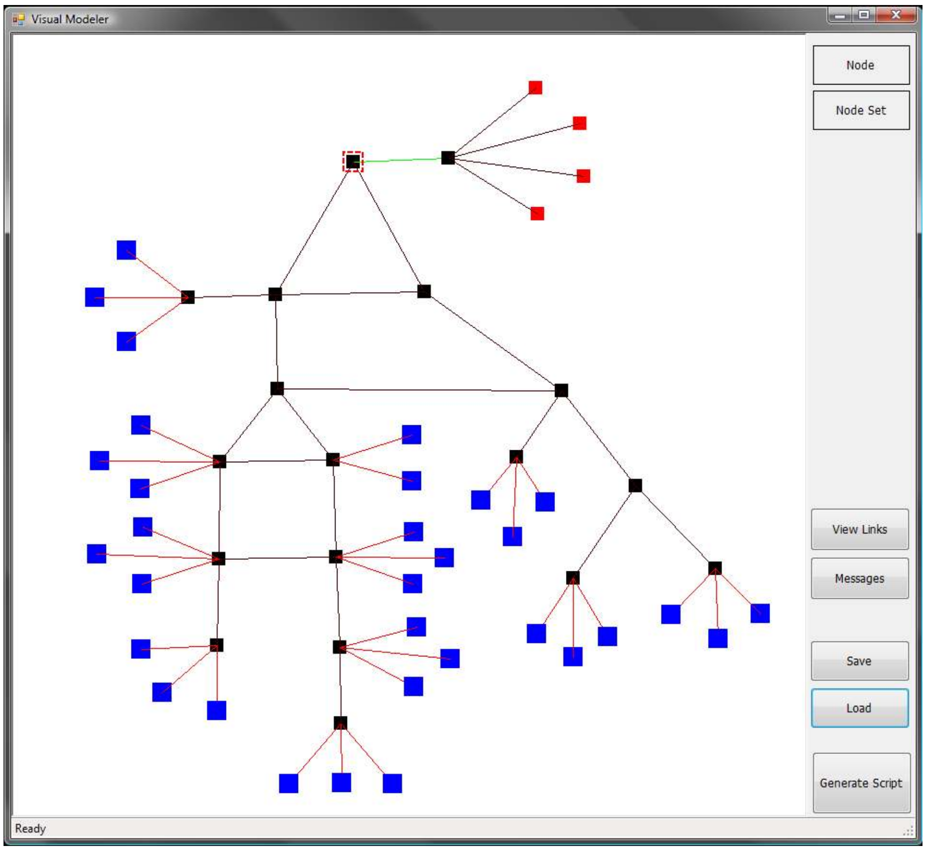


Figure 23 MMOG Network Model with 2400 Client and 4 Servers

The game messages send by the MMOG clients are similar to the ones that are described in the MMOG server bandwidth requirement analysis experiment. The MMOG clients move randomly in the virtual environment and as they move, they send their position updates to the MMOG servers with 50 bytes long data packets in every 1 to 3 seconds. Also, we have defined two additional game messages similar to the ones in the previous experiment. We have defined a *chat* message which can be 50 to 150 bytes long and sent by the MMOG clients in every 10 to 20 seconds.

Additionally, we have defined an *action* message which can be 75 to 100 bytes long and sent by the MMOG clients in every 5 to 15 seconds.

In the simulation models where there is more than one MMOG server, the virtual environment is divided into equal area regions and each region is associated with a server. We have defined a 50 bytes long connection message to be sent by a client, when it enters a new region hosted by a different server. Therefore, as the clients move randomly in the virtual environment, if they enter a new region, a connection message is sent by the client to the server associated with the new region.

For this experiment we have defined a virtual environment of size 400 x 400. We have configured the MMOG servers to use a Euclidean distance area-of-interest algorithm with 8 units of interest circle radius. Each simulation is run for 30 minutes of simulation time.

5.4.2 Results

During the simulations we have collected statistics regarding the number of packets sent by the clients and the server, the number of packets received by the servers, the number of connection messages sent by clients. Table 8 summarizes the results obtained from the first simulation model, which has a single MMOG server.

Table 8 Single Server Configuration Results for Experiment 3

	Server	Clients (Total)	Single Client
Packets Received	2870965	NA	NA
Bytes Received	173613654	NA	NA
Packets Sent:	11265333	2871001	1196.250417
Bytes Sent:	681322108	173616010	72340.00417

In the case where the whole virtual environment is hosted by a single server the server receives messages from clients at an average rate of 0.092 MB per second. On the other hand, the server sends messages to the clients at an average rate of 0.361

MB per second. In this case, the average number of bytes sent by a client is 0.039 KB per second.

Table 9 2 Server Configuration Results for Experiment 3

	Server 1	Server 2	Clients (Total)	Single Client
Packets Received:	1429318	1463246	NA	NA
Bytes Received:	86340754	88398087	NA	NA
Packets Sent:	5389191	5764235	2892623	1205.259583
Bytes Sent:	326000303	348744350	174742353	72809.31375
Connection Packets:	NA	NA	22115	9.214583333

Table 9 shows the results of the simulation model with two MMOG servers. In the case where the virtual environment is hosted by two servers, the servers receive messages from clients with an average rate of 0.046 MB per second whereas the average rate of the messages sent by the servers is 0.179 MB per second. On the other hand, a single client sends messages to the servers with an average rate of 0.039 KB per second. The average number of connection messages sent by a single client is 9.2. This means, an average of 0.25 bytes per second overhead is introduced in this configuration.

Table 10 3 Server Configuration Results for Experiment 3

	Server 1	Server 2	Server 3	Clients (Total)	Single Client
Packets Received:	788637	1322347	797087	NA	NA
Bytes Received:	47599383	79801962	48107845	NA	NA
Packets Sent:	2492743	6305001	2522738	2908912	1212.046667
Bytes Sent:	150703283	381281843	152439571	175512334	73130.13917
Connection Packets:	NA	NA	NA	38861	16.19208333

Table 10 shows the results obtained from the simulation model with three MMOG servers. In this case the average download and upload rate of the MMOG servers are 0.031 MB/s and 0.121 MB/s respectively. The average upload rate of a single client is 0.040 KB/s. The average overhead caused by the connection messages is 0.44 bytes per second.

Table 11 4 Server Configuration Results for Experiment 3

	Server 1	Server 2	Server 3	Server 4	Clients (Total)	Single Client
Pck. Rcv.:	513084	950132	949340	509626	NA	NA
Bytes Rcv.:	30930410	57311117	57268836	30783309	NA	NA
Pck. Sent:	1379515	4274244	4200727	1376650	2924346	1218.4775
Bytes Sent:	83265754	258358737	253939367	83205816	176296300	73456.79167
Connection Packets:	NA	NA	NA	NA	53841	22.43375

Table 11 summarizes the results obtained from the simulation model with four MMOG servers. In this simulation model, the average download and upload rate of the MMOG servers are 0.023 MB/s and 0.09 MB/s respectively. The average upload rate of a single client is 0.040 KB/s. The average overhead introduced by the connection messages is 0.62 bytes per second.

The simulation results show that, by dividing the virtual environment into regions and assigning a single server for each region can greatly decrease the bandwidth requirement and therefore the workload on the servers. For example, in single server configuration the bandwidth requirement of the server for upload is about 0.361 MB/s whereas in four server configuration this number is 0.09 MB/s. It is clear that, dividing the virtual environment will introduce some overhead in the communication because of connection and initialization messages. However, as it can be seen from the simulations the overhead remains insignificant in configurations with reasonable amount of servers.

CHAPTER 6

CONCLUSION & FUTURE WORK

This thesis study presents a network simulation tool for simulating typical client-server architecture based MMOG networks. The proposed tool (*GTNetS*-MMOG) can be used to model and simulate large scale MMOG networks which consist of several thousands of MMOG clients. For visually modeling the MMOG networks a network modeler tool (*Visual Modeler*) is also presented.

The network simulation tool presented in this thesis study is based on the *GTNetS* network simulator. For allowing the simulation of typical MMOG networks the *GTNetS* network simulator is extended with new application and message types. For simulating typical MMOG servers, an *MMOG Server* application type is introduced in *GTNetS*. Also, for simulating the typical MMOG clients, an *MMOG Client* application type is introduced. Furthermore, an infrastructure which allows sending of custom data structures with real data is added to the network simulation environment. The *MMOG Client* application can connect to the *MMOG Server* application using TCP protocol and can send messages with defined sizes and intervals. The *GTNetS-MMOG* allows collection of various statistics regarding the MMOG network. To allow users to define their simulation models in script files, a script parser is also added to the *GTNetS-MMOG*.

MMOG networks are massive in size. Designing a typical MMOG network with thousands of clients is a subtle task and could be time consuming. A network modeler tool which allows visually modeling of the MMOG networks is also implemented in this study. The *Visual Modeler* tool provides functionalities which simplify the modeling of large scale MMOG networks such as adding a number of nodes to the network topology with a single operation. The *Visual Modeler* tool also allows saving and loading the simulation models.

The use of the *GTNetS-MMOG* network simulation tool as well as the *Visual Modeler* tool is demonstrated by example simulations. We have used the *GTNetS-MMOG* tool to measure the effect of the interest circle radius on the network traffic in a typical MMOG network. Moreover, the effect of increasing the client count in an MMOG network to the bandwidth of the MMOG server is also observed by a simulation example. Finally we have conducted an experiment which shows the effect of dividing the virtual environment into regions that are hosted by separate servers on the overall network traffic.

In this thesis study we have provided a network simulation tool which can be used to model and simulate typical MMOG networks. We believe it is important to use simulation tools for testing, analyzing and verifying the design of the game during the development process. Since the MMOG development is an expensive process, testing and verifying features like the game network protocol at the very early phases of the development process could decrease the development costs. Also the overall quality of the service provided by the game is expected to increase, since various features of the game would be tested and verified by the simulations.

As a future work, the *GTNetS-MMOG* could be extended to support parallel distributed simulations. Since MMOG networks are massive in size, simulating large scale networks is crucial for an MMOG network simulator. We have managed to simulate networks with 8000 nodes with using *GTNetS-MMOG*. Although, this size is reasonable for most of the MMOG's, simulating higher number of nodes could be required by some MMOGs. Simulating higher number of nodes would require extensive computational power and memory requirements. Therefore, larger scale simulations should be done in a distributed manner.

Also the Visual Modeler tool can be improved. Apart from the user interface improvements, a feature which allows the users to define delays for the routers can be added. Since, adding delays to the routers would allow more realistic modeling of the MMOG network topology. Additionally the parser module of the GTNetS-MMOG can be extended to support additional simulation settings such as the packet color of the nodes, animation update interval and etc. Improving the parser module would decrease the amount of code needed to be written for a standard simulation.

REFERENCES

[1] Voig, Inc., “MMOGData.VOIG.com | Online Data About Online Games,” 2008. [Online]. Available: http://mmogdata.voig.com/. [Accessed: 6.27.2008].

[2] Woodcock, B. S., “An Analysis of MMOG Subscription Growth,” Apr, 2008. [Online]. Available: http://www.mmogchart.com/. [Accessed: 6.25.2008].

[3] DFC Intelligence, “DFC Online Game Market Forecast 2007,” 2007. [Online]. Available: http://www.dfcint.com/index.php. [Accessed: 6.11.2008].

[4] Zona, Inc., Executive Summary Consulting, Inc., “The Challenges of MMOG Development,” Digital Game Developer, Nov, 2002. [Online]. Available: http://www.digitalgamedeveloper.com/2002/11_nov/features/dlmmogd112602.htm. [Accessed: 6.12.2008].

[5] Assiotis M., and Tzanov, V. 2006. “A distributed architecture for MMORPG,” In Proceedings of *5th ACM SIGCOMM Workshop on Network and System Support For Games* (Singapore, October 30 - 31, 2006). NetGames'06. ACM, New York, NY, 4.

[6] Fujimoto, R. M., Perumalla, K.S. and Riley, G. F., 2006, “Network Simulation,” Synthesis Lectures on Communication Networks 2006 1:1, 1-72.

[7] Burray, B., “GTNetS – Home”. [Online]. Available: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/. [Accessed: 7.24.2008].

[8] Blizzard Entertainment, Inc., “World of Warcraft,” 2008. [Online] Available: http://www.worldofwarcraft.com/index.xml. [Accessed: 7.20.2008].

[9] “Blizzard Entertainment’s World of Warcraft: The Burning Crusade Surpasses…”, Apr 11, 2008. [Online]. Available: Reuters, http://www.reuters.com/article/pressRelease/idUS95187+11-Apr-2008+PRN20080411. [Accessed: 7.16.2008].

[10] YMIR Co. Ltd., “Metin 2”, [Online]. Available: http://www.metin2.org/. [Accessed: 7.23.2008].

[11] Ducheneaut, N., Yee, N., Nickell, E., and Moore, R. J. 2006. “"Alone together?": exploring the social dynamics of massively multiplayer online games,” In Proceedings of the *SIGCHI Conference on Human Factors in Computing Systems* (Montréal, Québec, Canada, April 22 - 27, 2006). CHI '06. ACM, New York, NY, 407-416.

[12] Vorderer, P. and Bryant, J. 2006. “Playing Video Games: Motives, Responses, and Consequences (Lea's Communication Series)”. Lawrence Erlbaum Associates, Inc.

[13] Tay, V. 2005. “Massively Multiplayer Online Game (MMOG) - A Proposed Approach for Military Application”. In Proceedings of the *2005 International Conference on Cyberworlds* (November 23 - 25, 2005). CW. IEEE Computer Society, Washington, DC, 396-400.

[14] “Maze War”, in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Mazewar. [Accessed: 6.22.2008].

[15] “ARPANET”, in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/ARPANET. [Accessed: 6.26.2008].

[16] “MUD”, in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/MUD. [Accessed: 7.3.2008].

[17] Kent, S., “The History of Massively Multi-player Online Games”. [Online]. Available: http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_9485_9488%5E9563%5E9599%5E9793,00.html. [Accessed: 7.7.2008].

[18] Koster, R. “Online World Timeline,” Feb 20, 2002. [Online]. Available: http://www.raphkoster.com/gaming/mudtimeline.shtml. [Accessed: 7.7.2008].

[19] Toth, V. T., “MUD 1 Home Site,” 2007. [Online]. Available: http://www.british-legends.com/. [Accessed: 7.23.2008].

[20] Near Death Studios, Inc., “Meridian 59 Official Web Site,” 2007. [Online]. Available: http://meridian59.neardeathstudios.com/. [Accessed: 7.2.2008].

[21] “Meridian 59”, in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Meridian_59. [Accessed: 7.2.2008].

[22] Electronic Arts, Inc., “Ultima Online”. [Online]. Available: http://www.uoherald.com/. [Accessed: 7.6.2008].

[23] NCsoft, “Lineage”. [Online]. Available: http://www.lineage.com/. [Accessed: 7.7.2008].

[24] Sony Online Entertainment, “EverQuest”. [Online]. Available: http://everquest.station.sony.com/. [Accessed: 7.8.2008].

[25] NCSoft, “Guild Wars”. [Online]. Available: http://www.guildwars.com/. [Accessed: 7.13.2008].

[26] Playnet Inc., “World War II Online,” 2008. [Online]. Available: http://www.wwiionline.com/. [Accessed: 7.13.2008].

[27] Sony Online Entertainment, “The Agency”. [Online]. Available: http://theagency.station.sony.com/. [Accessed: 7.13.2008].

[28] O2 Online Entertainment, “Mankind,” 2005. [Online]. Available: http://www.mankind.net/. [Accessed: 7.13.2008].

[29] Kru Interactive Inc., “Shattered Galaxy”, 2006. [Online]. Available: http://www.sgalaxy.com/. [Accessed: 7.13.2008].

[30] Linden Research, Inc., “Second Life,” 2008. [Online]. Available: http://secondlife.com/. [Accessed: 7.13.2008].

[31] El Rhalibi, A. and Merabti, M. 2005. “Agents-based modeling for a peer-to-peer MMOG architecture”. Comput. Entertain. 3, 2 (Apr. 2005), 3-3.

[32] Cronin, E., Filstrup, B. and Kurc, A., “A Distributed Multiplayer Game Server System,” 2001. EECS589 Course Project Report, University of Michigan.

[33] Van Den Bossche, B., et al., 2006. “A platform for dynamic microcell redeployment in massively multiplayer online games”. In Proceedings of the *2006 international Workshop on Network and Operating Systems Support For Digital Audio and Video* (Newport, Rhode Island, November 22 - 23, 2006). NOSSDAV '06. ACM, New York, NY, 1-6.

[34] Chan, L., Yong, J., Bai, J., Leong, B., and Tan, R. 2007. “Hydra: a massively-multiplayer peer-to-peer architecture for the game developer”. In Proceedings of the *6th ACM SIGCOMM Workshop on Network and System Support For Games* (Melbourne, Australia, September 19 - 20, 2007). NetGames '07. ACM, New York, NY, 37-42.

[35] Knutsson, B., Honghui Lu, Wei Xu, Hopkins, B., “Peer-to-peer support for massively multiplayer games”, INFOCOM 2004. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies* , vol.1, no., pp.-107, 7-11 March 2004

[36] Iimura, T., Hazeyama, H., and Kadobayashi, Y. 2004. “Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games”. In Proceedings of *3rd ACM SIGCOMM Workshop on Network and System Support For Games* (Portland, Oregon, USA, August 30 - 30, 2004). NetGames '04. ACM, New York, NY, 116-120.

[37] Yu, A. and Vuong, S. T. 2005. “MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games”. In

Proceedings of the *International Workshop on Network and Operating Systems Support For Digital Audio and Video* (Stevenson, Washington, USA, June 13 - 14, 2005). NOSSDAV '05. ACM, New York, NY, 99-104.

[38] Hampel, T., Bopp, T., and Hinn, R. 2006. “A peer-to-peer architecture for massive multiplayer online games”. In Proceedings of *5th ACM SIGCOMM Workshop on Network and System Support For Games* (Singapore, October 30 - 31, 2006). NetGames '06. ACM, New York, NY, 48.

[39] Bharambe, A., Pang, J., and Seshan, S. 2006. “Colyseus: a distributed architecture for online multiplayer games”. In Proceedings of the *3rd Conference on 3rd Symposium on Networked Systems Design & Implementation - Volume 3* (San Jose, CA, May 08 - 10, 2006). USENIX Association, Berkeley, CA, 12-12.

[40] Kim, K., Yeom, I. And Lee J. 2004. “HYMS: A Hybrid MMOG Server Architecture”. IEICE TRANSACTIONS on Information and Systems Vol.E87-D No.12 pp.2706-2713.

[41] Yoick Entertainment, “Outback Online”. [Online]. Available: http://www.yoick.com/outback/. [Accessed: 7.17.2008].

[42] “Earth & Beyond”, in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Earth_&_Beyond. [Accessed: 7.12.2008].

[43] Spencer, R., “The Tools Of The Trade,” 2005. [Online]. Available: IGDA Online Games Quarterly, The Technology Issue, V.1, Issue 3, http://www.igda.org/online/quarterly/1_3/persistentspencer.php. [Accessed: 7.1.2008].

[44] Miller, R., “Extended Outages for World of Warcraft,” 2005. [Online]. Available: http://news.netcraft.com/archives/2005/03/23/extended_outages_for_world_of_warcraft.html. [Accessed: 7.3.2008].

[45] Fritsch, T., Ritter, H., and Schiller, J. 2005. “The effect of latency and network limitations on MMORPGs: a field study of everquest2*”*. In Proceedings of *4th ACM SIGCOMM Workshop on Network and System Support For Games* (Hawthorne, NY, October 10 - 11, 2005). NetGames '05. ACM, New York, NY, 1-9.

[46] J. Smed, T. Kaukoranta, H. Hakonen, “Aspects of networking in multiplayer computer games”, In Proceedings of the *International Conference on Application and Development of Computer Games in the 21st Century*, 2001, pp. 74-81.

[47] Boulanger, J., Kienzle, J., and Verbrugge, C. 2006. “Comparing interest management algorithms for massively multiplayer games”. In Proceedings of *5th ACM SIGCOMM Workshop on Network and System Support For Games* (Singapore, October 30 - 31, 2006). NetGames '06. ACM, New York, NY, 6.

[48] Shankar, A. Udaya, *Discrete-Event Simulation,* 1991.

[49] Riley, G. F. 2003. "The Georgia Tech Network Simulator". In Proceedings of the *ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research* (Karlsruhe, Germany, August 25 - 27, 2003). MoMeTools '03. ACM, New York, NY, 5-12

[50] Riley, G. F., Ammar, M. H., and Fujimoto, R. 2000. "Stateless Routing in Network Simulations". In Proceedings of the *8th international Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (August 29 - September 01, 2000). MASCOTS. IEEE Computer Society, Washington, DC, 524.

[51] "Ns-2", in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Ns-2. [Accessed: 7.18.2008].

[52] Riley, G. F. 2003. "Simulation of large scale networks II: large-scale network simulations with GTNetS". In Proceedings of the *35th Conference on Winter Simulation: Driving innovation* (New Orleans, Louisiana, December 07 - 10, 2003). Winter Simulation Conference, 676-684.

[53] Luc Hogie, P. B. and Guinand, F. 2006. "An overview of MANET's simulation". Electronic Notes in Theoretical Computer Science, 150:81-101.

[54] "The Network Simulator - ns-2". [Online]. Available: http://www.isi.edu/nsnam/ns/. [Accessed: 7.15.2008].

[55] Altman, E. And Jimenez, T., Dec 4, 2003, "NS Simulator for beginners", Lecture notes at Univ. de Los Andes, Merida, Venezuela and ESSI, Sophia-Antipolis, France.

[56] "OTcl". [Online]. Available: http://otcl-tclcl.sourceforge.net/otcl/. [Accessed: 7.16.2008].

[57] "TclCL". [Online]. Available: http://otcl-tclcl.sourceforge.net/tclcl/. [Accessed: 7.24.2008].

[58] Riley, G. F., Ammar, M. H., Fujimoto, R. M., Park, A., Perumalla, K., and Xu, D. 2004. "A federated approach to distributed network simulation". ACM Trans. Model. Comput. Simul. 14, 2 (Apr. 2004), 116-148.

[59] OPNET Technologies, Inc., "OPNet Modeler". [Online]. Available: http://www.opnet.com/solutions/network_rd/modeler.html. [Accessed: 7.15.2008].

[60] Chen, K., Huang, P., Huang, C., and Lei, C. 2005. "Game traffic analysis: an MMORPG perspective". In Proceedings of the *International Workshop on Network*

and Operating Systems Support for Digital Audio and Video (Stevenson, Washington, USA, June 13 - 14, 2005). NOSSDAV '05. ACM, New York, NY, 19-24.

[61] P. Svoboda, W. Karner, and M. Rupp. 2007, “Traffic analysis and modeling for World of Warcraft”. IEEE International Conference on Communications, pp. 1612--1617, 24-28 June 2007.

[62] Jung, Y., et al., 2004, “VENUS: The Online Game Simulator Using Massively Virtual Clients”. AsiaSim 2004, Korea, October 2004, pp. 589-596

[63] “Uniform distribution (continuous)”, in Wikipedia, The Free Encyclopedia. [Online] Available: http://en.wikipedia.org/wiki/Uniform_distribution_(continuous). [Accessed: 8.1.2008].

[64] “Exponential distribution”, in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Exponential_distribution. [Accessed: 8.1.2008].

[65] “Pareto distribution”, in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Pareto_distribution. [Accessed: 8.1.2008].

[66] “Weibull distribution”, in Wikipedia, The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Weibull_distribution. [Accessed: 8.1.2008].

[67] Guzman, J., “Boost Spirit Home”. [Online]. Available: http://spirit.sourceforge.net/. [Accessed: 7.27.2008].

[68] Dawes, B., Abrahams D. and Rivera R., “Boost C++ Libraries”, [Online]. Available: http://www.boost.org/. [Accessed: 7.27.2008].