Abstract
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. 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.

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 [7], [7]. 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 [7]. 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 costs more than other multiplayer games [7].

One of the most important and challenging part of the MMOG development process is the design of the network architecture of the game. At any time, thousands of players can be online in a typical MMOG. 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 players is by no means an easy task.

In this study, we have developed a network simulation tool for MMOGs which can be used to model and simulate MMOG networks. Network simulation is used for various aspects of research and development of networks, including protocol analysis & verification, evaluating the network architectures and testing network scenarios [7]. Use of a network simulation tool can significantly help developer improve the quality of the network architecture and decrease the development time.

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 would improve the quality and decrease cost of 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.

2. MMOGS

2.1. Background & Characteristics

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 [7] which is one of the most popular MMOGs [7], [7] has achieved one million concurrent players in China at 11 April 2008.
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 are played by 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.

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).

2.2. Architecture of MMOGs

Nearly all of the existing commercial MMOGs have client-server architecture [7]. 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 [7]. 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 with MMOG Server 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.

To be able to deal with 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 [7]. There are many possibilities for distributing the responsibilities of the servers. One of the possible configurations is shown in the Figure 1.

In this sample server configuration, the server responsibilities split into group 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. Database servers store the game data, client avatar data and other configuration data related with the game. This server group, referred as MMOG Server farm, acts as a single server in the MMOG network model. MMOGs may have multiple copies of this server farms. For example World of Warcraft has North America, Europe and Asia servers [7]. 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 [7]. 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. For that reason, there have been a number of proposals on P2P architectures for MMOGs [7], [7]. There are also some hybrid architectures proposed [7]. 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 [7], which is an MMOG currently being developed by Yoick, may be
the first commercial P2P based MMOG.

2.3. **MMOG Development Challenges**

MMOGs are significantly more difficult and expensive to develop than the traditional video games [7]. 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 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 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 [7].

Operational and maintenance costs can be even higher than the development costs. The cost of maintaining a huge bandwidth for the operation of thousands of simultaneous player connections makes up the largest share in operational and maintenance costs [7]. Because of the high development cost of MMOG, 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.

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 section, an introduction of network simulation is given, and discrete event network simulation is described. This section also contains a survey of some of the commonly used Network Simulators. Finally, we present the rationale for choosing GTNetS (Georgia Tech Network Simulator) as our simulation environment.
3.1. Introduction

In MMOGs, the proper design of the network architecture design is 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 heterogeneous game network over the Internet is a challenge that MMOG developers face.

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. Some of these aspects are 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.

- Distributed Architecture for MMOG Servers: Network simulation methods can be used to justify the game server architecture. 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 tool.

- 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 [7]. The performance of different AOI algorithms can be measured and improved by using network simulation tool. The affect of changing parameters of the AOI algorithm on the game network traffic can also be observed. Therefore, it is possible to determine an optimal algorithm with optimal parameters for the MMOG.

- 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.
3.2. Discrete Event Simulation

Nearly all currently available network simulators are based on discrete event simulation for modeling the network behaviors [7]. In discrete event simulation, the operation of the simulation system is represented by the state changes. These state changes are caused by discrete events that are sorted with a chronological order. In a mathematical representation, the operation of the simulation system can be expressed as the following sequence:

\[
<s_0 , ( e_0, t_0 ) , s_1 , ( e_1, t_1 ) , s_2 , ( e_2, t_2 ) ... >
\]

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 \( s_1 \). Each event is assumed to happen instantly, meaning that they take 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 state of the system is only important at discrete points in time. The system state changes between those discrete points are ignored [7]. 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 [7]. The sending device starts to send data at some time \( T \) and it is received at the receiving node at time \( T + \Delta T \). It is clear that the state of the actual physical network varies during the time interval \( \Delta T \). However, the discrete event simulator can safely ignore these state changes.

3.3. Network Simulators

Extensive research on network simulators, leads us to GTNetS, NS-2, and OPNet. In this subsection, we will give a survey of them.

The Georgia Tech Network Simulator (GTNetS) [7] is a full featured network simulation environment that allows researchers interested in computer networks to study the behavior of moderate to large scale networks, under a variety of conditions [7]. GTNetS has support for a large number of applications and protocols including IEEE 802.3, IEEE 802.11 and Bluetooth. GTNetS also supports distributed simulations using a federated simulation approach [7]. 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 uses dynamic NiV-Vector routing method to reduce the memory footprint of the simulation. Parallelization ability and the NiV-Vector routing method used, grants GTNetS a high scalability.

The ns-2 (Network Simulator 2) is a widely used discrete event simulator [7]. It is developed and maintained by the Information Science Institute (ISI) at the University of Southern California and supported by DARPA and NSF. Ns-2 has considerable support for TCP, routing and multicast protocols over both wired and wireless (local and satellite) networks [7]. Large number of extensions for ns-2 can be found on the internet.

OPNet (Optimized Network Engineering Tools) [7] is a commercial discrete event network simulation environment. It supports analytical and hybrid simulation methods. Moreover, it supports an extensive list of protocols and enables it’s 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.

4. Related Work

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 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 [7]. This research 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 & 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 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 out that the TCP protocol produces a significant overhead because of TCP/IP headers and acknowledgements. They have argued that use of TCP protocol for MMORPGs can be considered as overkill.

Another similar study about traffic analysis of MMOGs has been done by Philipp Svoboda et.al. [7]. In their study, they have analyzed network traffic of the popular MMOG World of Warcraft [7]. 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.

YungWoo Jung et al. [7] 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, which are controlled from a central station, are implemented in a way that the game server can’t distinguish them from actual clients. 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 2 shows the block diagram of the VENUS system.

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 MMOG
5. A TOOL FOR NETWORK SIMULATION OF MASSIVELY MULTIPLAYER ONLINE GAMES

5.1. Requirements

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. To be able to simulate a 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 [7] [7], having TCP support is a required feature of network simulation tool for MMOGs.

5.2. 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 simulation of MMOG networks. Furthermore, we have implemented a modeler tool that allows visual modeling. 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 model in the simulation environment and running the simulation, the simulation environment reports the results to the user. The overall architecture of the MMOG simulation tool is given in the Figure 3.

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.

5.2.1. GTNetS and MMOG Extensions

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

*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.

**MMOG Extensions to the GTNetS.** In order to model MMOG networks, we added new applications and data messages to the GTNetS. The extensions that we have provided can be used to 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 4.

We have implemented two application objects in GTNetS, which are **MMOG Server** and **MMOG Client**. Our **MMOG Server** and **MMOG Client** implementations use TCP for communication. Additionally, we have 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 **MMOG Server** and **MMOG Client** as well as **MMOG Data Packet** are given below.

- MMOG Server: allows incoming connections from **MMOG Client** application with a configurable port. As the clients connect, **MMOG Server**

![Figure 4: GTNetS MMOG Extensions](image-url)
Server gives a unique id to the clients and stores them in a structure for later references.

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

In Figure 5, area of interest of the red colored player 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. 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, 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 sends to the **MMOG Clients** within the area of interest of the sender 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 generat-
ing random waypoints in a defined virtual area. 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. 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 as needed. 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 use of random distributions. We have implemented support for following random distributions:

- **Uniform**: A constant probability distribution function, i.e. all the values in the distribution interval are equally probable. Figure 6 shows the probability density function and cumulative distribution function of a continuous uniform distribution.

- **Exponential**: A continuous probability distribution function whose density function is given by

  \[ f(x) = ae^{-ax} \]  \hspace{1cm} (1)

  where \( a > 0 \) for \( x > 0 \), and \( f(x) = 0 \) for \( x \leq 0 \). Figure 7 shows the probability density function and cumulative distribution function of an exponential distribution.
Pareto: For a random variable $X$ with Pareto distribution, the probability that $X$ is greater than some number $x$ is given by

$$Pr(X > x) = \left(\frac{x}{x_m}\right)^{-k}$$

for all $x \geq x_m$, where $x_m$ is the (necessarily positive) minimum possible value of $X$, and $k$ is a positive parameter. Figure 8 shows the probability density function and cumulative distribution function of a Pareto distribution.

Weibull: A continuous probability distribution whose probability density function is given as $f(x; k; \lambda) = \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(x/\lambda\right)^k}$. The probability density function and the cumulative distribution function of a Weibull distribution is given in Figure 9.

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

- 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 GT-NetS. MMOG Client and MMOG Server communicate with using this message structure. The structure of the MMOG Data Packet is given in the Figure 10.

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.

5.2.2. Visual Modeler

In GTNetS, network simulations are written in C++ programming language. In small scale, C++ 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. Therefore, we have implemented a Visual Modeler tool for allowing the user to easily create and model MMOG networks without writing C++ code. Visual Modeler tool have been implemented with C# programming language and Microsoft Visual Studio 2005 IDE.

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. 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.

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 a thousand 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.

In *Visual Modeler* tool, the network links between the nodes can be cre-
ated 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: Figure 12, (A) shows a duplex link created between two nodes. 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 12, (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 12, (C)). There are no links between the nodes in a node group.

Visual Modeler tool allows defining custom game messages. A game message can be defined with two parameters, namely message size and 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.

Saving/loading of a simulation model is 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.
5.2.3. The Parser

We have implemented a parser module for the GTNetS-MMOG. The parser module parses GTNetS-MMOG scripts and creates the simulation environment.

We have used the Spirit parser framework which is a part of Boost C++ Libraries. Spirit is an object oriented recursive descent parser framework implemented using template meta-programming techniques. Spirit tries to mimic 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.

6. EXPERIMENTS & RESULTS

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

6.1. Simulation Environment

The network simulations described in this section 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.

6.2. AOI Algorithm Performance Evaluation

The effect of the area-of-interest (AOI) algorithm’s 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.
6.2.1. 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 [7]. We have used the Visual Modeler tool to model the network. Figure 13 shows the network topology in the Visual Modeler tool.

In the Figure 13, 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 mes-
The network simulations took an average of 4.5 hours to complete on the simulation computer. The simulations in which the radius of the interest circle is smaller took slightly less time than the simulations that have larger radius of interest circle. During the simulations, we have collected statistics about the MMOG Server latency, the packets send by the server, clients and the packets received by the server. The simulation results are summarized in the Table 1.

From the simulation results, we have 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 14.

The MMOG server and client bandwidth requirements can be calculated directly from these results. Table 2 shows the calculated average bandwidth requirements for MMOG server and MMOG client.

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.
Table 1: AOI Algorithm Simulation Results

<table>
<thead>
<tr>
<th>Radius of Interest Circle</th>
<th>5</th>
<th>11</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Server Latency</td>
<td>0.0191</td>
<td>0.0191</td>
<td>0.0191</td>
</tr>
<tr>
<td>Worst Server Latency</td>
<td>0.0401412</td>
<td>2.7099</td>
<td>2.70671</td>
</tr>
<tr>
<td>Average Server Latency</td>
<td>0.029232</td>
<td>0.0292292</td>
<td>0.0292242</td>
</tr>
<tr>
<td>Server RCV (Packets)</td>
<td>5384258</td>
<td>5385530</td>
<td>5386353</td>
</tr>
<tr>
<td>Server RCV (Bytes)</td>
<td>269212900</td>
<td>269286150</td>
<td>269323200</td>
</tr>
<tr>
<td>Server SND (Packets)</td>
<td>35703313</td>
<td>176558826</td>
<td>418639786</td>
</tr>
<tr>
<td>Server SND (Bytes)</td>
<td>1785165650</td>
<td>8828267850</td>
<td>20932433900</td>
</tr>
<tr>
<td>Client SND (Packets)</td>
<td>5384323</td>
<td>5385782</td>
<td>386531</td>
</tr>
<tr>
<td>Client SND (Bytes)</td>
<td>269216150</td>
<td>269289100</td>
<td>269326550</td>
</tr>
</tbody>
</table>

Table 2: AOI Algorithm Simulation: Server and Client Bandwidth Requirements

<table>
<thead>
<tr>
<th>Interest Circle Radius</th>
<th>5</th>
<th>11</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Bandwidth (B/s)</td>
<td>991758</td>
<td>4904593</td>
<td>11629130</td>
</tr>
<tr>
<td>Client Bandwidth (B/s)</td>
<td>24.92</td>
<td>24.93</td>
<td>24.93</td>
</tr>
</tbody>
</table>
of the clients when the radius of the interest circle is 17 units.

6.3. 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.

6.3.1. 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 13). By changing the node counts in the node groups, we adjusted the MMOG client count on the network. We have modeled four MMOG networks with client sizes 960, 2880, 4800, and 6720 respectively.

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.

6.3.2. Results

During the simulations we have collected statistics about the messages sent and received by the MMOG server and clients. Table 3 summarizes the collected statistics. Note that the client send statistics are for average values for a single MMOG client.

Calculated bandwidth requirement values from these results are given in the Figure 15. 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.
Table 3: MMOG Server Bandwidth Requirement Analysis Results

<table>
<thead>
<tr>
<th>Client Count</th>
<th>960</th>
<th>2880</th>
<th>4880</th>
<th>6720</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server RCV (Packets)</td>
<td>1263027</td>
<td>3789908</td>
<td>6309364</td>
<td>8816587</td>
</tr>
<tr>
<td>Server RCV (Bytes)</td>
<td>98832631</td>
<td>296452097</td>
<td>494130891</td>
<td>691673342</td>
</tr>
<tr>
<td>Server SND (Packets)</td>
<td>3489314</td>
<td>31521597</td>
<td>86209742</td>
<td>171784391</td>
</tr>
<tr>
<td>Server SND (Bytes)</td>
<td>273126945</td>
<td>2466443921</td>
<td>6753706833</td>
<td>13479137198</td>
</tr>
<tr>
<td>Client SND (Packets)</td>
<td>1316</td>
<td>1316</td>
<td>1294</td>
<td>1316</td>
</tr>
<tr>
<td>Client SND (Bytes)</td>
<td>102952</td>
<td>102936</td>
<td>101221</td>
<td>102928</td>
</tr>
</tbody>
</table>

Figure 15: MMOG Server and Client Bandwidth Requirements
6.4. 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.

6.4.1. Simulation Model

In order to observe the performance of the overall MMOG network with different virtual environment partitioning, we modeled four network simulation models with 2400 clients and 18 routers. 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 16 shows the MMOG network model with four MMOG servers in Visual Modeler tool.

The game messages send by the MMOG clients are similar to the ones that are described in the MMOG server bandwidth requirement analysis.
Table 4: Single Server Configuration Results for Experiment 3

<table>
<thead>
<tr>
<th></th>
<th>Server</th>
<th>Clients (Total)</th>
<th>Single Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets Received:</td>
<td>2870965</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Bytes Received:</td>
<td>173613654</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Packets Sent:</td>
<td>11265333</td>
<td>2871001</td>
<td>1196.250417</td>
</tr>
<tr>
<td>Bytes Sent:</td>
<td>681322108</td>
<td>173616010</td>
<td>72340.00417</td>
</tr>
</tbody>
</table>

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 into a new region, a connection message is sent by the client to the server associated with the new region.

6.4.2. 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, and the number of connection messages sent by clients. Table 4 summarizes the results obtained from the first simulation model, which has a single MMOG server.

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 5 shows the results of the simulation model with two MMOG servers.
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 and 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 6 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 7 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
### Table 6: Three Server Configuration Results for Experiment 3

<table>
<thead>
<tr>
<th></th>
<th>Server 1</th>
<th>Server 2</th>
<th>Server 3</th>
<th>Clients (Total)</th>
<th>Single Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pck. Rcv.:</td>
<td>788637</td>
<td>1322347</td>
<td>797087</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Bytes Rcv.:</td>
<td>47599383</td>
<td>79801962</td>
<td>48107845</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Pck. Sent:</td>
<td>2492743</td>
<td>6305001</td>
<td>2522738</td>
<td>2908912</td>
<td>1212.04</td>
</tr>
<tr>
<td>Bytes Sent:</td>
<td>150703283</td>
<td>381281843</td>
<td>152439571</td>
<td>175512334</td>
<td>73130.13</td>
</tr>
<tr>
<td>Con. Pck.:</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>38861</td>
<td>16.19</td>
</tr>
</tbody>
</table>

### Table 7: Four Server Configuration Results for Experiment 3

<table>
<thead>
<tr>
<th></th>
<th>Serv. 1</th>
<th>Serv. 2</th>
<th>Serv. 3</th>
<th>Serv. 4</th>
<th>Clients (Total)</th>
<th>Single Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pck. Rcv.:</td>
<td>513084</td>
<td>950132</td>
<td>949340</td>
<td>509626</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Bytes Rcv.:</td>
<td>30930410</td>
<td>57311117</td>
<td>57268836</td>
<td>30783309</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Pck. Sent:</td>
<td>1379515</td>
<td>4274244</td>
<td>4200727</td>
<td>1376650</td>
<td>2924346</td>
<td>1218.4</td>
</tr>
<tr>
<td>Bytes Sent:</td>
<td>83265754</td>
<td>258358737</td>
<td>253939367</td>
<td>83205816</td>
<td>176296300</td>
<td>73456.7</td>
</tr>
<tr>
<td>Con. Pck.:</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>53841</td>
<td>22.4</td>
</tr>
</tbody>
</table>
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.

7. CONCLUSION

This study presents a network simulation tool for simulating 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 study is based on the GT-NetS network simulator. For allowing the simulation of typical MMOG networks, the GTNetS network simulator is extended with new application and message types. 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 define their simulation models in script files, a script parser is also added to the GTNetS-MMOG.

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 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 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.
References


Chen, K., Huang, P., Huang, C., and Lei, C. 2005. ”Game traffic analysis: an MMORPG perspective”. In Proceedings of the International