Performance Monitoring on Networked Virtual Environments

Christos Bouras, Eri Giannaka

***Abstract*— As networked virtual environments gain increasing interest and acceptance in the field of Internet applications, a lot of effort is drawn in the optimization of their performance, which will result in the wide adaptation of this evolving mean of communication. This paper focuses on the performance monitoring of such a networked virtual environment, called EVE, in order to identify the elements that degrade its network performance.**

***Index Terms*— networked virtual environments, performance monitoring, simulation**

I. Introduction

The inherent need of humans to communicate acted as the moving force for the formation, expansion and wide adoption of the Internet. The need for communication, collaboration and learning from distance, resulted in the evolvement of the primitive services originally offered (i.e. e-mail) to advanced functionalities, which offer a high sense of realism to the user, forming a reality, the so-called virtual reality. Virtual environments were, at first, used in the field of entertainment, through computer games. The innovation was then pointed out to the fact that a real place could be simulated, using 3D technology, where the user could navigate and interact with 3D objects as in real life.

Even though virtual environments were first introduced as stand alone applications, which could run on a single computer, the promising functionalities of this new form of representation and interaction as well as the familiarity of the users with it drew increased research interest. This fact resulted in virtual reality to be viewed as the solution for achieving communication and collaboration between scattered users, both in the field of entertainment as well as in the newborn field of distance learning. This led to the creation of networked virtual environments (NVEs) [14].

Manuscript received March 5, 2004.

Christos Bouras is with the Research Academic Computer Technology Institute, Patras, Greece (phone: +30-2610-960375, fax: +30-2610-960358, e-mail: bouras@cti.gr) and the Computer Engineer and Informatics Department, University of Patras, Greece (phone: +30-2610-996951, fax: +30-2610-969016, e-mail: bouras@ceid.upatras.gr).

Eri Giannaka, is with the Research Academic Computer Technology Institute, Patras, Greece phone: +30-2610-960380, fax: +30-2610-960358, e-mail: giannaka@cti.gr) and the Computer Engineer and Informatics Department, University of Patras, Greece.

Many platforms which adopted the virtual reality technology were implemented and networked virtual environments started to be enhanced with additional functionality, as text and audio chat, streaming media support, application sharing, etc. However, these enhanced features are demanding both in terms of system as well as in terms of network resources. In particular, the distribution of the shared 3D objects of the virtual worlds and of the changes and modifications on them, the concurrent transmission of audio and video as well as the effort for consistency on the scattered users' views create resource overhead and cause delays that can be critical for the consistency and reliability of the networked environment. Therefore, the big challenge is perceived to be the optimization of the network performance and the reliability of the networked environments.

In this paper we describe such a networked virtual environment, called EVE [1], which is intended to be used not only for the support of multi-user virtual environments, but also for the provision of advanced services which can efficiently support distance learning and educational purposes. In addition, we focus on the aspects of the system that should be taken into consideration in order the virtual environment to be efficient and reliable, as well as the parameters that should be monitored and simulated in order to identify the time consuming and demanding components of this platform. By identifying these parameters, it becomes more feasible to define and determine ways or architectural models, under which the platform could present an optimized networked performance.

The performance monitoring and evaluation of such a networked virtual environment can be performed using tools called, networked simulators. These tools have a long history, almost equal to that of the Internet, since there has always been an increased interest on how efficiently the newborn applications operated over the network. There are many simulators currently available, each of which is characterized of some basic and some advanced functionalities, which can serve all kinds of applications, in respect with the nature of each of these applications.

The remainder of this paper is organized as follows. We will begin by describing EVE's architecture, in order to present the communication model that is a basic factor for the network performance. In chapter III we will present the tool that we have selected in order to approach the network behavior of EVE and the reasons that led to its adaptation. Chapter IV is engaged with the simulation model that was

used for the experiments on EVE. These experiments are described in chapter V. Finally we conclude with the results that are raised from the experiments conducted on EVE and the future work that we have planned in order to get an efficient and reliable environment.

II. EVE Architecture

EVE's architecture is based on a client-multi server platform model. The current form of EVE constitutes a simple structure, which allows and supports the basic functionality that the platform is intended to offer [2], [3].

One of the most basic considerations of Networked Virtual Environments is the maintenance of the consistency of the users' view, which is translated in efficient and reliable updates for the modifications that take place in the Virtual Worlds. Based on this notation, the structure of the server side for EVE was directed to the decrement of the load for the processes that actualize and transmit the updates of the Virtual Worlds. Therefore, the server side architecture comprises of two basic components, one main server, the Message Server, who is mainly responsible for the maintenance of the consistency and the application servers, which host and support additional services offered by the platform. In particular EVE comprises two application servers, a chat and an audio server.

In order the users' client to communicate with EVE's servers and have access to the provided functionalities it needs a web browser, a VRML browser, a VRML client, a main client, a chat client and an audio client.

The network communication of EVE, alike its architecture, is focused on providing the available functionality at the best possible performance. Therefore, for the transmission of the packets and the achievement of the communication of the connected clients with the host servers (message server, audio server and chat server) as well as for the server-to-server communication, there are two types of communication supported. Each of these types is found to be optimum for certain kinds of messages. Thus, we categorized the messages exchanged in the EVE communication platform in three basic categories: a) the messages related with the initialization of the virtual world and the initial connection of a client to a server as well as the messages exchanged between the servers of the platform, b) the position messages that are related with the avatars' position in the virtual environment and finally c) the important messages, which correspond to messages that are vital for the consistency of the networked virtual environment. For simplicity reasons, we consider as important messages all messages except for the position messages.

III. NS-2 Simulator

As mentioned in [4], NS2 is an object-oriented, discrete event driven network simulator, which was designed and implemented at the University of Berkley. NS2 is written in C++ and Otcl and is proven to be primarily useful for simulating local and wide area networks. In particular, this network simulator implements network protocols as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. In addition NS implements efficiently multicast and some MAC level protocols for LAN simulations.

From the information provided for NS2 in [4], the comparison among JSim, SSFNet and NS conducted in [8], as well as from the results of [5], [6] it came up that ns-2 represents a powerful solution for the simulation of networked virtual environments, as its design covers a wide range of characteristics and options that could be met in a variety of such applications.

In particular, as mentioned above, ns-2 supports and simulates efficiently a wide range of protocols, as well as multiple kinds of networks as local networks, mobile and satellite networking along with various kinds of routing as unicast and multicast. Consequently, this network simulator has the ability to simulate successfully networked virtual environments that may combine one or more of the above mentioned characteristics. In addition, the object orientation of C++ along with the programming simplicity of Tcl make ns-2 one of the most powerful tools for the simulation of the network performance of a wide range of applications.

Ns-2 was selected for the performance monitoring on EVE and the its evaluation mainly because of the fact that it is able to efficiently and successfully support both TCP and UDP communication, which, as mentioned above, are adopted by EVE's communication model.

IV. Simulation Model For EVE

The first step for the successful simulation of the platform is to define the parameters that need to be measured in order to get accurate and valid results about the system's performance.

A. Parameters Measured

As mentioned, the main scope of the work presented is to evaluate the performance of EVE by conducting experiments in order to detect the components of the system that could degrade the overall performance. These components could be described by parameters, which should be measured, in order to estimate, evaluate and get accurate results about the network performance of EVE.

Throughput: This parameter represents the amount of data moved successfully from one place to another in a given time period and can provide valuable conclusions on the efficiency of the platform regarding the delivery of the vital information.

Drop Rate: This parameter represents the average rate that the system disallows messages due to system failures or network congestions. In particular, this parameter can lead to conclusions on the reliability that the platform offers for the transmission of the events.

B. Simulation Topology

In the current version of EVE, there is one available Message Server and two Application Servers, a Chat and an Audio Server. Regarding the number of the clients connected, an efficient lower level can be calculated for the case that the platform supports only one virtual world.

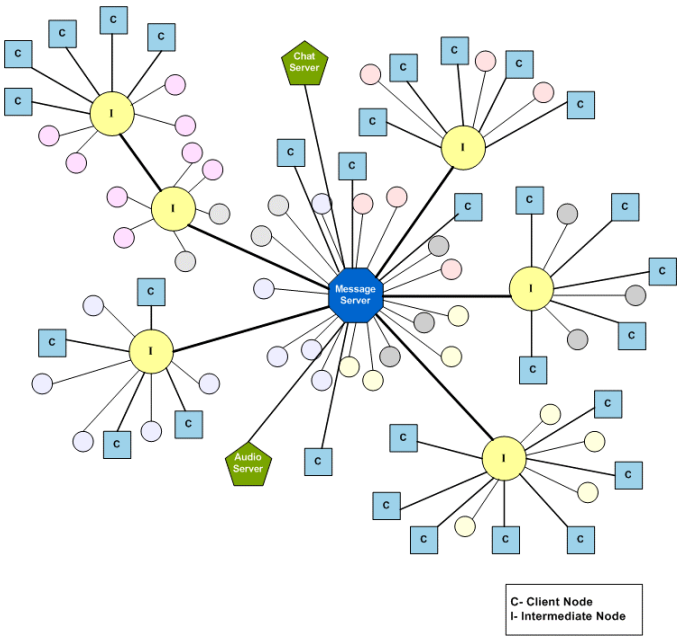

Fig. 1: Topology used for the experiments

In this case the number of the clients can vary from one to seventeen, which is translated in educational terms in sixteen students and one tutor. For reasons of approaching real case scenarios we define the number of the connected clients to thirty. The topology that has been used for the conducting of the experiments is shown in Fig. 1.

C. Service and Protocol description

The basic step for the realization of the experiments is the translation of the communication model of EVE to the programming structure that the ns-2 simulator adopts. Based to the communication model that was described above the following paragraphs present the connections that could take place during a learning scenario in regard to some basic events that can take place within the frame of EVE along with the modifications that had to be made for the successful evaluation of the platform's performance.

1) User enters the Virtual Environment

Initially, each client generates a connection request to the message server, which adopts the TCP protocol for terms of reliability and safety. Therefore, a TCP agent is attached to the Client's node and a TCPSink agent is attached to the Message Server for each of the connected clients. In addition, the Message Server, each time that a connection request is received, must notify the client and the application servers that the new user is added. For this communication the TCP protocol is again adopted. Therefore, for achieving this functionality, we programmed a new TcpSink-like agent, which triggers a new TCP agent which transmits a message to the corresponding nodes. As mentioned above, the communication between the application servers and the message server is established for synchronization reasons each time that a client decides to join or leave a virtual world. In this case the Message Server sends a TCP message. For security and consistency reasons, the application servers send back an acknowledgement each time that such a message is received. For this reason a TCPSink agent is attached to both Chat and Audio Servers' nodes.

2) User interacts with the Virtual Environment

However the TCP protocol is also used for the exchange of the "important" messages that take place in the virtual environment. For these cases, the Message Server, each time it receives such a message, it retransmits it to all connected clients but not to the application servers.

3) User moves in the environment

For the user's movement in the Virtual Environment the UDP communication is adopted. Therefore, we set a UDP agent to each of the client nodes for transmitting the UDP messages that are connected to the avatars position within the environment. Each time that a UDP message is received by the Message Server it must be retransmitted to all connected clients. Therefore we create a new UDP-like agent, which is attached to the message server node for each client and which triggers that transmission of the received UDP message to all connected clients.

4) User interacts with the application servers

Since the application servers receive the "connection" message for a client, a direct communication between them is established, without the interference of the Message Server. Thus, the processing load in the Message Server, which is mainly responsible for the maintaining of the consistency, is decreased by an important factor. In particular, the Client-Chat Server communication adopts once again the TCP protocol. Therefore a TCP agent is attached to the Client. Regarding the Chat Server, every time that a TCP message is received by one of the clients, it must be retransmitted to all connected clients. For this reason we programmed a TCPSink-like agent, which emulates the described functionality. Finally, we attach a TCPSink agent to all clients in order to send an acknowledgement of receiving the messages sent by the Chat Server.

As it happens in the case of the Chat Server, similarly, for the Audio Server, after the Message Server receives the "connection" message, a direct communication is established between this server and the respective client. For the Audio Communication, the H323 protocol is adopted. Therefore, we implement H323 traffic and attach a UDP agent to all connected clients. Each time that the Audio Server receives a H323 message by one of the clients, it retransmits it to all connected clients through a new UDP-like agent.

5) Users leaves the Virtual Environment

When a user decides to exit from the Virtual Environment a

message a TCP message is sent to the Message Server, which in turn notifies the application servers for the withdrawal, so as to destroy the direct connection, which has been established between them.

In the Table 1 the functions, the communication that takes place and the corresponding protocols are presented. MS stands for Message Server, AS for Application Servers, AC for Audio Server, CS for Chat Server and C for the client.

Function	**Communication-Protocols**		
Enter VE	1. C-**TCP**-MS	2. MS-**TCP**-AS/C	3. AS-**TCPSink**-MS
Interact with VE	1. C-**TCP**-MS	2. MS-**TCP**-AS/C	
Move in VE	1. C-**UDP**-MS	2. MS-**UDP**-C	
Text Chat in VE	1. C-**TCP**-CS	2. CS-**TCP**-C	
Audio Chat in VE	1. C-**UDP**-AS	2. AS-**UDP**-C	
Leave VE	1. C-**TCP**-MS	2. MS-**TCP**-AS/C	3. AS-**TCPSink**-MS

Table 1: Functions, communication and protocols used

V. Experiments

There are two types of experiments conducted in order to monitor the performance that the platform presents. The basic factor that was modified in the results is the background traffic that is created in some additional nodes, which simulates the Internet traffic. It should be mentioned that during the experiments we assume that the connected clients do not run other types of applications.

Regarding the topology we must mention that the Message Server, which is presented as the red hexagon in Figure 1, is directly connected to the Application Servers, which are presented as green boxes and with some of the clients which are presented in dark red. The Message Server is also connected to some nodes, which we call Intermediate nodes and are presented in yellow color. Between the Message server and the Intermediate nodes as well as between some of the Intermediate nodes we create some background traffic that fills a line of 10 MB.

In the first case we create a background traffic that fills in a line of 10 MB with traffic that its average throughput is 7 Mbps.

In the second case we consider a heavier background traffic that fills a line of 10 MB with traffic that its average throughput is 8 Mbps.

For both of these cases we try to measure the average throughput as well as the average drop rate for three types of connections. These types are the following:

1) A client is directly connected to the message server
2) A clients is one hop away from the message server
3) A client is two hops away from the message server.

A. Background Traffic A

This subsection is dedicated to the results of the experiments conducted for the performance of the platform under Background Traffic A, considering that the connected clients have available a line of 2 MB, which is shot of additional traffic load.

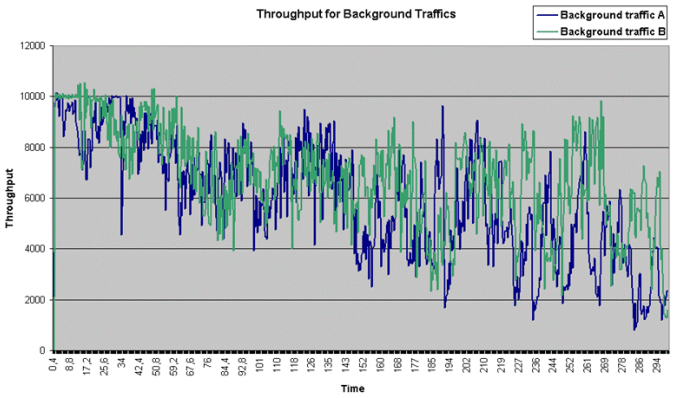

Fig. 2: Throughput of the Background Traffics

1) Throughput under Background Traffic A

During the simulation of this experiment we measured the throughput of the traffic that the Virtual Environment creates during the simulated period of time, which is set to 5 minutes. This traffic corresponds to both TCP messages and UDP messages, which have been forwarded by the message server each time that an update takes place either to the state of the world or to the position of the avatars.

The results of this experiment are displayed in the following figure.

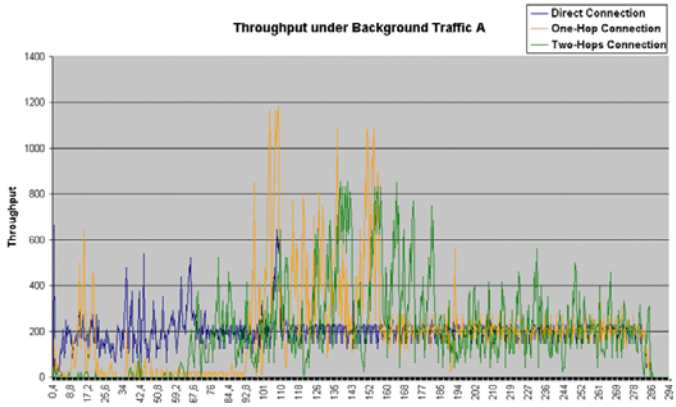

Fig. 3: Throughput under Background Traffic A

The throughput for the client that is directly connected to the server reaches an average amount of 240 Kbps when for the client that is situated one hop away from the message server and is affected by the background traffic that takes place between the Message Server and the Intermediate node, the throughput reaches an average of 195 Kbps. Finally, the throughput of the client which is two hops away and is affected by the background traffic that takes place both between the Message server and the First Intermediate node as well as between the first Intermediate node and the second Intermediate node reaches an average of 145 Kbps. What is important to be noticed in the above figure is the distribution that the lines follow for the hops that are situated one or two hops away, the first 80 seconds. At that period of time, the throughput of the lines is very small, which indicates that only

a small amount of information manages to pass through the Intermediate lines. This notice will be explained by the drop rate that follows for the intermediate lines.

2) Drop Rate for Background Traffic A

This sub-section presents the results that came up from the experiments on the intermediate lines in order to monitor the drop rate percentage that takes place in these lines. This drop rate is presented in Fig. 4.

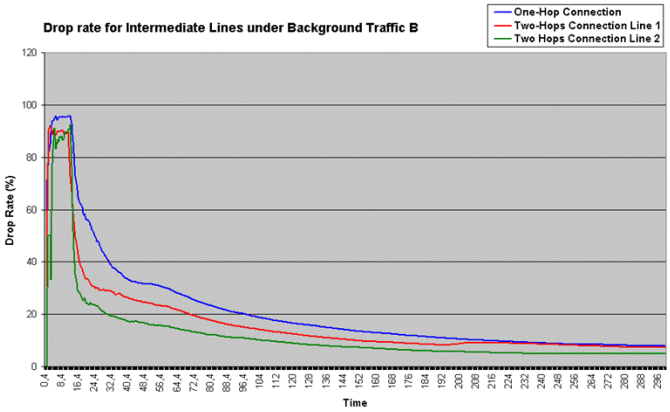

Fig. 4: Drop Rate for the Intermediate lines under Background Traffic A

From the Figure above, we notice that at the beginning of the simulation, in the Intermediate lines that are affected by the Background Traffic A, there is a drop rate which initially reaches a loss of more that 90% for both connections, that is one hop and two hops connections. This percentage is gradually reduced, until it reaches a percentage of 10%. It should be noticed that the shape of the lines for the drop rate agree with the throughput of the background traffic, which is presented in Fig.2. In particular, from Fig. 2 we can remark that at the first 80 seconds, the line is almost or totally filled because of the Background Traffic A. Therefore, only a small amount of packets manages to passes through the line, while the rest of the packets, nearly 90% are dropped.

This large amount of dropped packets could lead to the conclusion that the platform fails to support such a large number of users and both the consistency as well as the reliability and safety of the delivery is harmed. However, before dropping such a conclusion we should examine what part of this percentage corresponds to loss of “important” messages, so as to form a precise opinion on the harmed consistency. These percentages are displayed in Fig. 5.

From this figure we notice that the major part of the lost information corresponds to UDP messages, which, as mentioned earlier, are associated to the users’ movement within the virtual environments, and only a small percentage of lost information corresponds to the “important” TCP messages. However, the TCP messages, lost are retransmitted, the inconsistencies of the events that take place within the virtual world are minimized. It should be noted, that the retransmissions, lead to additional load for the platform, which can be observed in the unsmooth shape that the throughput follows for this traffic.

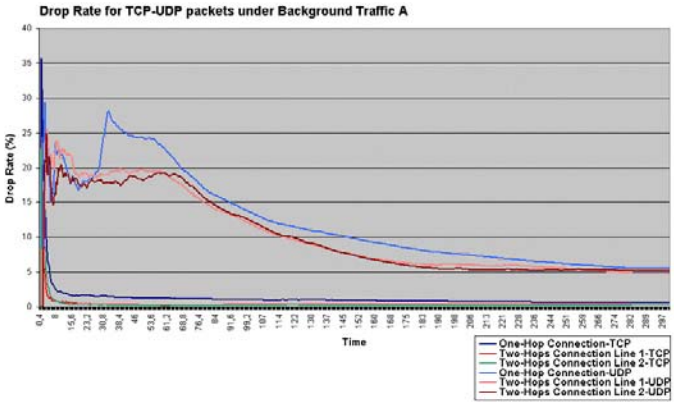

Fig. 5: Drop Rate for the Intermediate lines under Background Traffic A

Regarding the direct connection of the client to the Message Server, the drop rate is very small and achieves and optimized performance.

B. *Background Traffic B*

This subsection is dedicated to the results of the experiments conducted for the performance of the platform under Background Traffic B, considering again, for comparison reasons, that the connected clients have available a line of 2 MB, which does not suffer of any additional kind of traffic.

1) Throughput under Background Traffic B

In this case we consider a heavier background traffic that fills the line of 10 MB, which connects the Message Server to the Intermediate Nodes and the Intermediate Nodes to the clients. This background traffic has an average bandwidth of 8 Mbps.

From the experiments conducted from this type of traffic we notice that the throughput for the client that is directly connected to the server reaches an average amount of 100 Kbps when for the client that is situated one hop away from the message server reduced almost at approximately 70 Kbps. Finally, the throughput of the client which is two hops away and is affected by the background traffic that takes place both between the Message server and the First Intermediate node as well as between the first Intermediate node and the second Intermediate node is reduced to approximately 45 Kbps.

Similarly to the case of Background Traffic A, we notice in the above figure that the shape that the lines follow for the hops that are situated one or two hops away, the first 80 seconds indicates a very small value of the throughput. This fact indicates that only a small amount of information manages to pass through the Intermediate lines. This notice will be explained by the drop rate that follows for the intermediate lines.

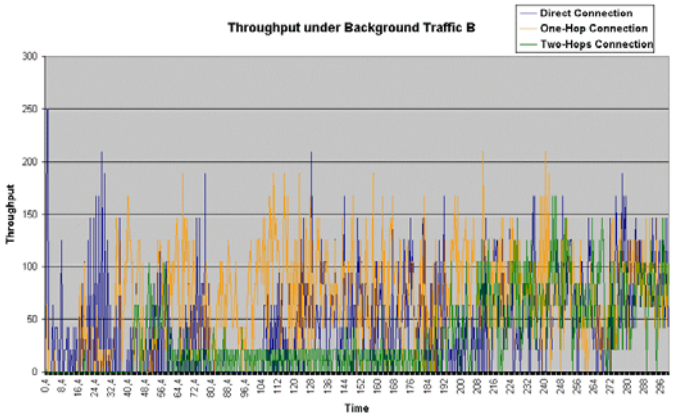

Fig. 6: Throughput under Background Traffic B

2) Drop Rate for Background Traffic B

This sub-section presents the results that came up from the experiments on the intermediate lines in order to monitor the drop rate percentage that takes place in these lines, under Background Traffic B. This drop rate is presented in Fig. 7.

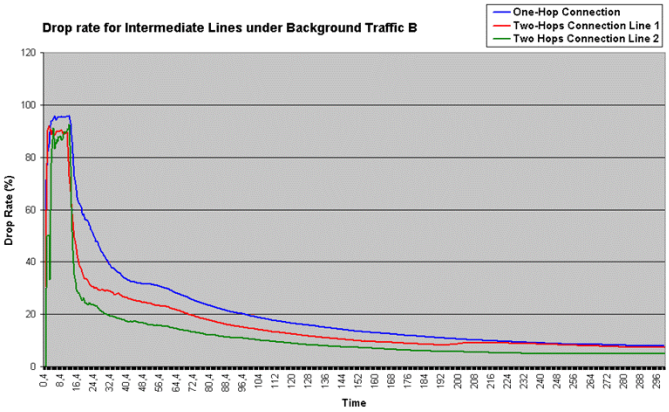

Fig. 7: Drop Rate for the Intermediate lines under Background Traffic B

From the Figure above, we notice that at the beginning of the simulation, in the Intermediate lines that are affected by the Background Traffic A, there is a drop rate which initially reaches a loss of more that 95% for both connections, that is one hop and two hops connections. This percentage is gradually reduced, until it reaches a percentage of 10%. It should be noticed that the shape of the lines for the drop rate also agrees with the shape of the throughput for Background Traffic B, which is presented in Fig.2.

As we did, in the case of Background Traffic A, in order to investigate the severity of this large amount of dropped packets, we should examine what percent of this amount corresponds to loss of "important" messages, so as to decide on the platforms' performance. These percentages are displayed in Fig. 8.

From this figure we notice that also in this case, the major part of the lost information corresponds to UDP messages, which, as mentioned earlier, are associated to the users' movement within the virtual environments, and only a small percentage of lost information corresponds to the "important" TCP messages.

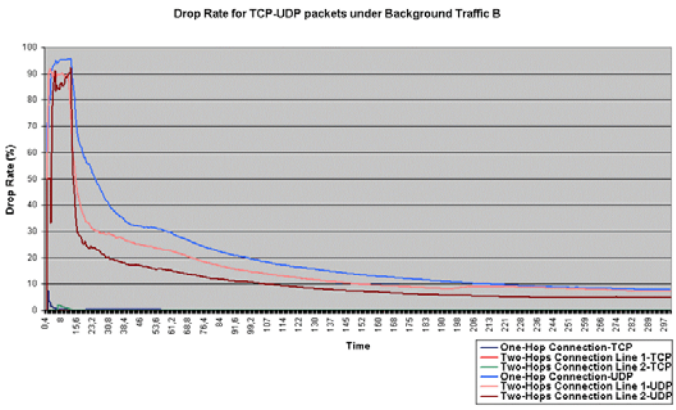

Fig. 8: Drop Rate for the Intermediate lines under Background Traffic B

Regarding the direct connection of the client to the Message Server, the drop rate, also in this case, is very small and achieves and optimized performance.

VI. Results

Comparing the results from these two experiments we notice that the Background Traffic B results to a severe loss of information in comparison to Background Traffic A. It should also be noted that for the direct connections one might have supposed that the throughput would fluctuate in the same range. However, the severe Background Traffic B results in loss of information from the clients to the Message Server, and thus the number of packets to be forwarded is reduced. It should be mentioned that the reason for not performing experiments to the Client-Message Server direction, is because in this case, a packet that will be lost will never get to the Message Server and consequently to the connected clients. Therefore, there are no alterations in the view of the world that the participants share.

In addition, from the experiments conducted, we ascertain that the platform seems to operate efficiently under Background Traffic A and for both cases the communication model of the platform ensures that the lost packets will not affect severely the consistency of the virtual worlds since there is only a small amount of "important" messages that need to be retransmitted.

However, it should also be mentioned that from the above figures and given the fact that the clients' link bandwidth to both the Intermediate nodes as well as to the Message Server is set to 2 MB, the Message Server seems unable to handle such a number of simultaneous users.

VII. Conclusion And Future Work

In its current form EVE is based on a simple architectural model that can support efficiently only a limited number of parallel virtual worlds and simultaneous users. However, during the primer implementation of the EVE platform, we took into account the possible need for extension due to increased demand and participation. The new architectural model uses the same rational as the original model, which

means that both the processing load as well as the communication load should be distributed regarding with the connections and requests [4].

The new model, which will be used for the extension of the EVE platform is displayed in Fig. 9 and is still based on two types of servers, the message server and the application servers. These servers could be the audio and chat server, as they are in the system, as well as some additional servers for advanced functionality.

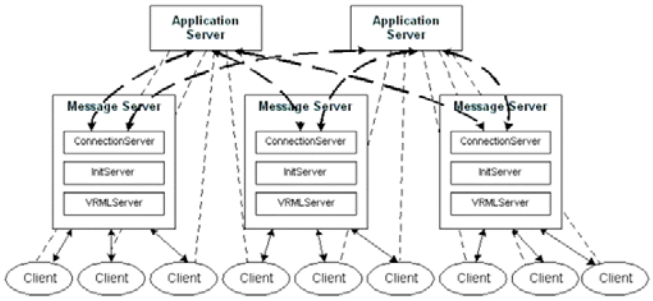

Fig. 9: Expanded architecture of EVE

The number of the message servers can be increased regarding the client connections and the processing load. Each message server shelters a certain number of virtual worlds. This number will result from the performance monitoring that we will conduct on EVE.

However, the main concept and innovation of this architectural model is the fact that each message server is a back up server for a number of the rest message servers. This implies that if a failure happens, the clients supported and hosted by the "damaged" message server can be distributed to the other servers of the system. In addition, each message server can be a client to another message server, when the processing load of the first one exceeds its hosting capabilities. The selection of the second message server, which will undertake the accessional tasks is selected with the following simple algorithm: the processing and network load of all available message servers is computed, in respect to the shared objects in each virtual worlds, the connections and instances of the virtual environments, and the one with the smaller load undertakes the tasks. In the case where the load is similar to all available clients, the "host" server is selected using the Round Robin algorithm.

Therefore, one of the basic next steps is to implement an extended version of EVE and perform additional simulation on it so as to estimate the extended architectural schema and extract conclusions on how the optimization of its performance could be achieved.

VIII. References

[1] EVE (Educational Virtual Environments) prototype, http://ouranos.ceid.upatras.gr/vr/.

[2] Bouras C., Tsiatsos Th., "Architectures and Protocols for Educational Virtual Environments", IEEE International Conference on Multimedia and Expo (ICME2001), Tokyo, Japan, August 22-25 2001, pp. 1107 - 1110

[3] NS by example, http://nile.wpi.edu/NS/

[4] Teixeira R.C, and Duarte O.C.M.B, "`Evaluating the Impact of the Communication System on Distributed Virtual Environments"', Multimedia Tools and Applications, Kluwer Academic Publishers, vol. 19, no. 3, pp. 259 -- 278, March 2003

[5] Euisook Hong, Dongman Lee, Kyungran Kang, "An Efficient Synchronization Mechanism Dynamically Adapting to the Network State for Networked Virtual Environments", Journal of KISS: Information Networking, 2003. 2.

[6] S.Bajaj, L.Breslau, D.Estrin, K.Fall, S.Floyd, et al., "Improving Simulation for Network Research, technical report", University of Southern California/Information Sciences Institute, CA, USA, 99-702b, 1999

[7] Comparison of Network Simulators Revised, http://www.ssfnet.org/Exchange/gallery/dumbbell/dumbbell-performance-May02.pdf

[8] Packetizer, H323 information site, http://www.packetizer.com/iptel/h323/

[9] C. Bouras, E. Giannaka, T. Tsiatsos, "Issues for the Performance Monitoring on Networked Virtual Environments", 7th ConTEL – International Conference on Telecommunications, Zagreb, Croatia, June 11 – 13 2003, pp. 725 - 728