, , 1–22 ()

A Platform for Distributed Virtual Environments

RENATA CRUZ TEIXEIRA renata@gta.ufrj.br

OTTO CARLOS M. B. DUARTE otto@gta.ufrj.br

Grupo de Teleinformática e Automação (GTA), COPPE/EE - Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro

Abstract. This paper describes and analyzes the performance of a platform for distributed virtual environments on the Internet. We present the principles that have guided the platform design, together with the architecture and the protocol specified for this platform. The main goals of this platform are: scalability, to enable its use by a large number of users, and flexibility, to allow for different virtual reality applications. We obtain performance measurements via simulation, varying the number of participants, the partition of the virtual environment, a participant's speed, and the size of the virtual world. Results show that the division of the environment improves the delay both to enter and to change groups, and reduces the number of messages exchanged. This partition allows both a participant's speed and the number of participants to increase, thereby ensuring a good interactivity.

Keywords: Distributed Virtual Environments, Multicast, Scalability

1. Introduction

Distributed virtual environments (DVE's) aim at providing users with a sense of realism by incorporating tri-dimensional graphics and stereo sound to create an immersive experience. In these environments, also called virtual worlds, geographically dispersed users can interact with one another and with the environment as in the real world. The integration of immersive graphics with the ability to have multiple users sharing information and manipulating objects in the environment makes virtual reality systems a powerful tool. These systems are being used for education and training, for engineering and design, for commerce, and for entertainment.

For all their potential, DVE's are difficult to implement correctly or effectively for a large number of users [18]. The main limitations to scale distributed virtual reality systems are the constraints due to the communication system [12]. A DVE must contend with data losses, limited bandwidth, and delay.

In these environments, each participant has a graphical representation, called an avatar. As a user moves within the virtual environment, his avatar's current state must be updated so that other users visualize the avatar in the correct location to prevent inconsistent representations of the virtual environment among different hosts. If a user picks up an object in the environment, other users need to be notified that the object is now being carried. Thus, each participant must continually send state updates to all the others that share the same environment, which causes a great volume of data exchange. Nevertheless, a participant does not interact with all others in the world simultaneously. For example, in a large DVE, it is rare for all the participants to stand in the same room. Instead, users tend to scatter into different sub-groups or teams, each located in a different part of the virtual world. Hence, exchanged messages may be filtered, defining a scope within which participants can interact with one another, this scope is called an interaction scope. This mechanism of dividing the environment into scopes is called data filtering.

Some proposals [14, 9, 2] were based on data filtering to optimize the use of communication system resources, translating each interaction scope into a multicast group. Since participants may move around the world, group partition does not remain static. Therefore, multicast protocols to support DVE's must allow dynamic group changes. Besides, the delay introduced by these group changes should be minimum, for the purpose of avoiding interrupting a user's perception of the virtual environment.

One of the pioneering distributed virtual environments was the SimNet (Simulator Networking) System [10] of the American Department of Defense (DoD). The SimNet is a distributed military virtual environment for training small units to fight as a team. An attempt to formally generalize and extend the SimNet protocol was called the DIS (Distributed Interactive Simulation) protocol, the IEEE 1278 standard [17]. The SimNet and the DIS, however, are unsuitable for large-scale multi-player virtual environments [14].

Research into large-scale virtual worlds has been growing significantly. Based on the DIS, the NPSNET-IV [14, 13, 11], developed by the Naval Postgraduate School, is a simulation system for military training. This system implements data filtering, partitioning the world into classes. Each class has an area-of-interest manager which is responsible for it. This reduces the computational load in hosts and the information exchanged through the network. DIVE (Distributed Interactive Virtual Environment) [9] considers a multicast group per world, hence there is no data filtering within a world. Communication for DIVE uses a reliable multicast protocol with negative acknowledgement (NACK). The Model, Architecture and System for Spatial Interaction in Virtual Environments (MASSIVE) [8, 7] system uses a spatial model for interaction among clients. Each object has an *aura* that determines the space within which interactions are possible. Two participants exchange state updates when their *auras* collide. SPLINE (Scalable Platform for Large Interactive Networked Environment) [2] splits the world into locales, which can be processed separately.

These systems adopt the mechanism of data filtering along with multicast protocols aiming at achieving more scalable DVE's. They also intend to limit the scope within which consistency control is performed, distributing this control among various servers. Unfortunately, documentation for those systems is not easily found. This, combined with our interest in the constraints related to the communication system, led us to propose a general platform that combines the main characteristics of those systems in order to measure the impact of DVE's on the communication system.

This paper describes the proposed platform for distributed virtual reality systems on the Internet, and assess its communication performance. This platform provides a consistency maintenance and distribution service for virtual reality applications. In order to allow for a larger number of users, the platform divides the participants into multicast groups. Performance measurements consider different requirements of virtual reality applications: the partition of the virtual environment, a participant's speed, the number of participants, and the size of the virtual environment. We analyze the delay both to enter the virtual world and to change groups, the number of group changes due to participant's movement, and the number of messages exchanged.

This paper proceeds as follows: section 2 describes platform design issues, the architecture, and the protocol proposed for it. Section 3 presents the simulation model and the infrastructure. Section 4 discusses simulation results, used to evaluate the impact of different requirements on platform behavior. Finally, section 5 concludes.

2. The Proposed Platform

The proposed platform uses transport protocols available on the Internet in order to support distributed virtual reality applications. The service provided by the platform is flexible, i.e., the platform is configured according to the application. The application provides the scene for the virtual environment, graphical representations of participants (avatars), and information about the environment's structure. Based on this information, the platform is responsible for all interactions in the virtual world, ensuring world consistency. Thus, for the application, data distribution and control of objects of the virtual environment are transparent.

In addition to consistency maintenance and concurrency control to access virtual-world objects, distributed virtual reality systems should enable some sort of communication to occur among the participants through multiple media, like audio and video. Multimedia communication enhances the sense of realism in virtual environments. Therefore, the architecture proposed for the platform is modular, divided into functions for direct interaction among participants (multimedia), concurrency control (interaction with the virtual world), and functions of the virtual environment (consistency control and state updates).

2.1. Platform Design

The platform design has two main goals: flexibility, to be suitable for various kinds of distributed virtual reality applications, and scalability, to allow for a large number of users without harm to the interactivity among them. To provide scalability, the platform divides participants into interaction scopes according to the structure defined by the application. These scopes are translated into multicast groups, and represent the domain of consistency control.

Consistency control is ensured inside each group by a special host called leader. The leader host is elected among ordinary participant hosts, gathering leader functions. Each interaction scope has exactly one leader. When a participant changes from an interaction scope to another, hence changing multicast groups, he performs a handoff. The leader is responsible for controlling all participant entrances, departures, and handoffs within his interaction scope. In addition, the leader controls access to objects of the virtual environment.

Division of participants into multicast groups reduces the communication system load by filtering the exchange of update messages. As participants move around the virtual world, their interaction scope and their multicast group change, therefore there is a demand for dynamical handoffs. This dynamism is crucial for a seamless perception of the environment, which could be interrupted when a handoff occurs. As a result, the delay to notify a handoff and to effectively change groups must be minimized.

A distributed data model is adopted, partitioning virtual-world data among its users. Each participant host maintains a copy of the database referring to its interaction scope, therefore this database is replicated among participants of the same scope. In order to maintain consistency among these replicas, the communication among group participants must be reliable. Multicast protocols, however, impose limitations on the number of users when reliability is needed [16]. Besides, reliable protocols introduce delays to ensure that packets are delivered in order and with no errors, hence these protocols are not suitable to transfer data that need to be delivered in real time [12]. Therefore, the proposed platform reduces reliable communication by dividing virtual-world operations into critical operations, which need reliability, and non-critical operations, which must be delivered in real time. Operations like entering the virtual environment, departing, and changing groups are classified as critical operations. On the other hand, state updates and multimedia interactions, for which the main constraint is the delay, are non-critical operations.

The platform uses three communication types with different features. Non-critical messages require minimum delay and are received by all of the participants within a scope. Critical operations demand reliable message exchanges, which may be unicasted (to deliver the database, for example) or multicasted (for example, when an avatar is sent to a group). In order to support all of these requirements, the use of three different transport protocols together with a network service that provides multicast delivery is needed. The IP Multicast [3] allows hosts to join and leave groups at any time, i.e., the membership in a group is dynamic. Therefore, it was used along with the UDP for the transmission of non-critical messages. For the critical multicast messages, the transport protocol employed was the SRM (Scalable Reliable Multicast) [6], a reliable multicast protocol with negative acknowledgments, which aims at scalability. TCP is used to exchange messages between two participants reliably.

In summary, scalability is achieved by the distributed consistency control, which is performed by the leaders, and by the reduction of reliable data exchange, performed by the division of operations.

2.2. Platform Architecture

The architecture was designed to be used by all of the participant hosts in the same way, i.e., all of the hosts performing the platform are uniform, even leaders. Thus, any participant may be elected leader during a session. The only distinction between a leader and an ordinary participant is that the former performs, besides regular functions, some control tasks. The platform architecture is presented in figure 1. The functions accomplished by leaders are in an independent module, and are only launched when a participant becomes a leader. These leader functions are carried out without awareness of the user, who interacts with the virtual world as any other participant attached to an ordinary host (non-leader).

The virtual-world division into scopes is defined by the application. These scopes do not vary during a session. So, given the initial position of a participant in the virtual environment, his scope and his leader can be easily determined. Data of the virtual environment are partially replicated in the database (DB), which contains the scene and avatar information. By updating this database, users modify the state of the virtual world.

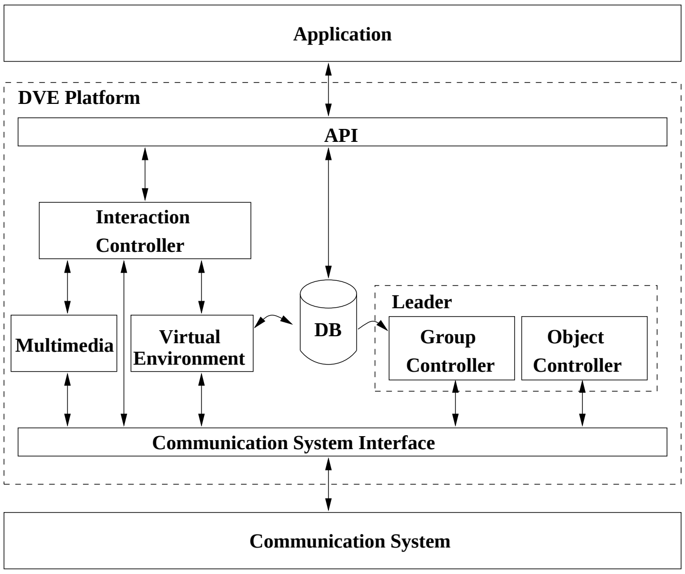


Figure 1. The proposed platform architecture.

The module virtual environment is responsible for maintaining information of the virtual environment updated. This module monitors the DB to verify the changes that must be sent to other participants, and receive update messages from them. This module may perform a dead-reckoning algorithm aiming at reducing the frequency of update messages. Dead reckoning [15] is a mechanism for movement prediction used to determine the position of other participants based on their previous movements. Thus, a participant does not need to send a message for every movement of the user. The algorithm must be chosen by the application considering the behavior of its avatars.

Interactions between users and with the environment are managed by the module interaction controller. This module is responsible for setting up a connection between the participants, so that they can interact directly through the multimedia module. This multimedia module provides direct communication through gesture, typed text, or voice. The interaction controller is also responsible for the concurrency control to access objects in the world, which it achieves by verifying object permissions.

Leader hosts, in addition to the regular modules, use two special modules: the group controller and the object controller. The group-controller module manages entrances, departures and handoffs within a leader interaction scope. The object-controller module has the access permissions to every object in a leader scope, thereby avoiding conflicts among participants accessing the same object.

2.3. Service and Protocol Description

In order to achieve a flexible platform, the protocol has to be general. It cannot assume characteristics that are specific to an application as, for example, the DIS protocol [17], which was specified for military applications. Thus, the protocol data units (PDU's) specified may be used by any kind of distributed virtual reality applications, including military applications.

The platform includes interactions both among users and with the virtual environment, and the control required to maintain world consistency. This paper's main focus is the control of information of the virtual environment in order to achieve consistency maintenance. This control includes functions that determine system scalability – world entrances and departures, state updates, and handoffs. Therefore, the protocol description presented here concentrates on these functions.

Table 1. Service Primitives.

Primitive	*Function*
VR_Con.Req	Request to enter the virtual world
VR_Con.Ind	Indication that a new avatar joined the scope
VR_Con.Conf	Notification that data of the virtual environment is available
VR_Dis.Req	Request to leave the virtual world
VR_Dis.Ind	Notification of a participant's departure
VR_Update.Req	Participant's movement must be update in the DB
VR_Update.Ind	Notification of an object's update

Service primitives presented in table 1 can be split into three groups according to their function: entrance in the virtual world, departure from the virtual world, and update. On the entrance phase the application module of the new participant sends a VR_Con.Req primitive, requesting the database, which contains information of the virtual environment. As a response, the application receives a VR_Con.Conf primitive notifying that the database is available and that the other members of the scope have received the new avatar. Other participants detect the entrance of the new avatar upon receipt of a VR_Con.Ind primitive. During a session, user movements will cause the transmission of VR_Update.Req primitives. These primitives will cause the receipt of VR_Update.Ind primitives. If the platform employs a dead-reckoning algorithm, update requests may be filtered, and hence they will not cause the transmission of a VR_Update.Ind. Primitives to leave the virtual world are VR_Dis.Req and VR_Dis.Ind. They notify the platform that the avatar must be removed from the environment. It is important to remark that there are no primitives to accomplish handoffs because group changes are performed without the awareness of the application.

PDU's are presented in table 2. Those used to perform the entrance in the virtual world are JR, JI, and JC. To leave the virtual world an LR PDU is issued. Handoffs are performed using entrance and departure PDU's, together with the FNL PDU, which notifies a participant host that it has to begin a handoff.

Table 2. Protocol Data Units.

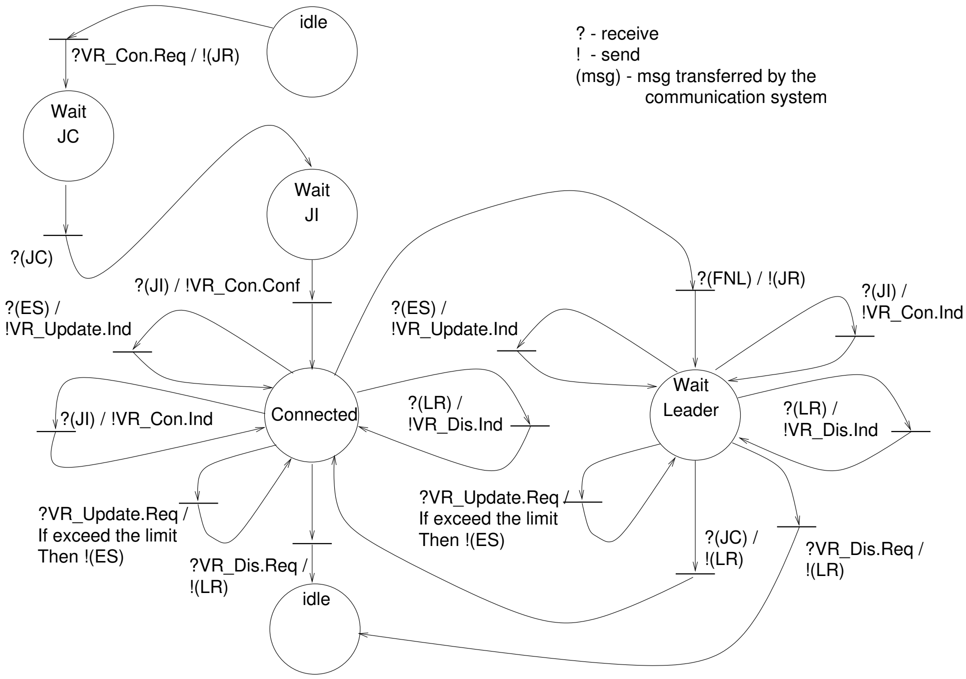
PDU	*Purpose*
JR	Request to join a group
JI	Indicate a new avatar joined the group
JC	Confirm a join request, by sending the DB
LR	Leave the group
FNL	Indicate that a participant should initiate a handoff
ES	Update an avatar's state

A Predicate-Action Petri Net [4] model for the protocol's behavior is shown in figure 2. Figure 2(a) presents the behavior of a regular participant, and figure 2(b) presents functions performed by leaders.

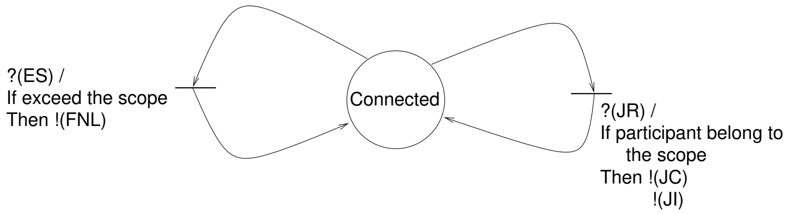
Each virtual-world participant is member of a multicast group. This group is determined according to its position in the virtual environment. All leaders belong to a multicast group of control, which has a known address. Thus, participant hosts send messages to this control address to discover which leader is responsible for the scope in which its avatar will be placed.

To begin a session, i.e., to enter the virtual world, the application must receive information about the scope in which it is entering, and must send its avatar to the other members of the scope. Thus, the application in the participant host issues a VR_Con.Req primitive to the platform, which then multicasts a JR PDU to the control address requesting to join a group. The JR PDU carries the initial position and the information of the new-participant avatar. Upon receipt of a JR PDU, all leaders verify whether the position requested belongs to its interaction scope. The leader responsible for this position answers the request by sending a JC PDU, which carries the DB containing the current state of its scope.

After delivering the DB, the leader multicasts to all members of its scope a JI PDU with information of the new participant. Upon receipt of a JI PDU with its avatar, the platform issues a VR_Con.Conf to inform the application that information of the virtual environment is updated and that its avatar was placed in the environment. Then, this participant can interact with the virtual world. When the



(a) Regular participant.



(b) Leader.

Figure 2. Protocol behavior modeled using Petri Nets.

other participants of the scope receive a JI PDU, the platform issues a VR_Con.Ind notifying the entrance of the new avatar.

After the entrance phase, participants update their states through VR_Update.Req and VR_Update.Ind primitives, which cause the exchange of multicast messages with ES PDU's in their interaction scope. An ES PDU reports avatar position and behavior information. The leader, as any other participant, receives every state update of all members of its scope. When it detects that a participant is crossing the boundaries of its scope, the leader sends an FNL PDU to that participant indicating that a handoff must be started. Thus, the participant multicasts a JR PDU and restarts the entrance phase. After receiving a JC PDU from its new leader, the participant leaves its old group by multicasting an LR PDU. It is important to remark that handoffs are managed by platform modules without the knowledge of the application.

To finish a session, a participant issues a VR_Dis.Req primitive. Upon receipt of this primitive, the platform multicasts an LR PDU to the participant's scope, requesting the removal of its avatar from the environment. Other participants of the scope receive a VR_Dis.Ind primitive.

3. Simulation

The platform performance was assessed via simulation. Simulations were performed using a topology similar to the Internet's structure, and included operations for consistency maintenance: world entrances and departures, handoffs, and state updates.

3.1. Simulation Infrastructure

Simulations are carried out by the *ns* (Network Simulator) [1, 5], which is a discrete-event simulator used in the networking research community. It provides an infrastructure for network protocol simulation, which has abstractions to nodes and links, and support for routing, including multicast routing. Moreover, it includes several models of common Internet protocols (e.g., TCP, UDP), plus several novel protocols targeted for eventual use in the Internet (e.g., reliable multicast).

Transport protocols used were the UDP, the TCP, and the SRM. Some adjustments were needed in the SRM protocol in order to fit platform requirements. In SRM, when a new user joins a session, the user will obtain the history of the session from the network [6]. On DVE's, however, when a user joins a group he only wishes to receive the current state of his interaction scope. The receipt of the history of the session would cause a large number of useless messages, and waste both communication system resources and processing time for these messages to be discarded by the new member. Therefore, the SRM module of *ns* was adapted not to send the history of the session to new group members.

3.2. Topology Generation

The network topology was created by the topology generator GT-ITM (Georgia Tech Internetwork Topology Models), aiming at graphs characteristics that resemble those of the Internet. The GT-ITM uses a hybrid generation approach [19], the Transit-Stub model, which allows the generation of topologies using a domain structure similar to that found in the Internet.

The generated graphs have several sub-networks (stubs) interconnected by cores (transit), which represent Internet backbones. Figure 3 shows the topology used in the simulations. This topology has 40 nodes – 8 in the transit plus 8 stubs with four nodes each. Link latencies vary between 0.3 and 3.8 ms, and a bandwidth of 2 Mbps is used. We assume that platform users are always in stub domains, and that there is only one user for each stub node. Therefore, in simulations using the topology presented the maximum number of participants simulated is 32.

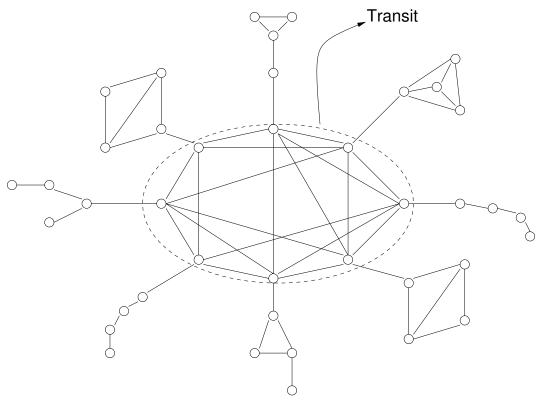


Figure 3. The topology used in the simulations.

3.3. Simulation Model

The platform depends on some characteristics and information defined by the application, which are translated into parameters of the simulation model. The application must set the database containing the initial scene of the virtual environment, this database is translated into a parameter with the initial size of the database. The avatar definition is converted into a parameter that express the size of an avatar. We assume that all avatars have the same size.

These parameters are used for scope joins and leaves. When a new user joins a group, the database is increased by the size of the new avatar. Similarly, when a participant leaves its scope, the size of its avatar is decreased from the database size.

In the simulations performed, the virtual environment is divided into fixed areas. The space of the virtual world, i.e., the number of possible positions remains, while the number of groups varies. Consequently, when the number of groups increases, the area of each group reduces. The virtual environment is divided among all leaders, so that there is no intersection among the areas of different leaders, and that the union of all areas results in the virtual world itself. Leaders are randomly chosen among all possible participants. All random values were generated using the random variables available in *ns*. The initial position of participants is randomly selected among all possible positions in the environment as well. There is no simulation of leader election during a session. Leaders are chosen in the beginning of a session, and are the first to enter the virtual world. They remain still, i.e., in the same group, during the whole session and leave the virtual world in the end of it, after all the other participants. Entrance and departure times of non-leader participants are randomly generated. Participant's movement around the world is associated with a probability (P_m). Thus, with probability P_m a participant moves one position ahead, i.e., increments its position by one unit. This probability is related to a participant's speed, thus, a faster participant should have a higher probability of movement. In the simulations performed, we consider that all participants have the same probability of movement. For each participant's movement, a state-update message is multicasted to its scope. When a participant's position cross the boundaries of its scope, its leader notifies the beginning of its handoff.

Simulations were performed based on this model, and using the topology described in section 3.2. Simulation parameters used to obtain the results are presented in table 3. The database initial size, which represents an interaction scope, is the largest parameter. A simple scope can be expressed by 10 kbytes, avatars are not complex and are represented by 1.5 kbytes, and update messages only contain behavior information and thus are represented with 100 bytes. The probability of movement adopted is 20 %, which stands for medium-speed avatars, like a person running. The platform was simulated with sessions of five minutes each.

Table 3. Parameters used for the simulation.

Parameters	*Values*
DB initial size	10 k bytes
Avatar size	1,5 k bytes
Update size	100 bytes
P_m	20 %

3.4. Measurements Performed

The measurements performed aimed at evaluating the behavior of the platform under requirements of different applications. In order to assess the platform we measured: delay both to enter the world and to change groups, the number of messages exchanged, and the number of handoffs in a session.

Delay to enter the virtual world is composed by:

1. Transmission of a join request (JR PDU) to the leaders by SRM

2. Waiting time due to the leader reaction to the request

3. Establishment of a TCP connection

4. Transmission of the database by the TCP

Any participant entering the virtual world is subjected to delays imposed by the items 1 and 3, regardless the system load. The delay introduced by the item 2, however, occurs when the leader responsible to answer the new participant is overloaded, i.e., it is answering the request of another participant. It is important to remark that the *ns* does not consider processing time. Therefore, the delay represented by the item 2 stands for queueing delay only. Delay related to item 4 depends on the size of the data to be transferred, varying with the size of the virtual environment.

Other aspects that influence the delay are the topology used and the position of the leader and of the new participant in this topology. The topology, as described in section 3.2, is formed by sub-networks interconnected by cores. The path between sub-nets can pass through one or more cores. The larger the number of cores between sub-nets, the larger the number of hops in the path, and hence the delay. The topology also determines link latency and bandwidth, features that influence the transfer delay. The position of both participant and leader hosts in the topology affect the delay because they determine the number of hops in the path.

Delay to perform a handoff is formed by the same elements of the entrance delay plus the delay for the leader to notify that a participant must begin a handoff. Therefore, the first element of the handoff delay is the delay to transfer an FNL PDU through TCP.

Another measurement was the number of messages exchanged by the platform. Messages are counted for each link they traverse. Thus, each message sent by a host (leader or participant) that generates more the one message on links – because it is a multicast message, its path has more than one hop, or it is segmented by TCP – is counted more than once. Messages were divided into two groups: control messages, which are the critical messages (JR, JC, JI, FNL, LR PDU's); and non-critical messages, update messages (ES PDU's).

Finally, we measured the number of handoffs. This measurement stands for the total number of handoffs performed by all of the participants together during a session. Each handoff introduces a delay that interrupts the perception of the virtual environment. Therefore, the larger the number of handoffs, the larger the number of interruptions. This measurement is topology independent, it depends only on the behavior and the distribution of the avatars in the virtual world, and how this world is partitioned into scopes. This measurement is also influenced by the number of users.

4. Platform Performance

The platform was analyzed considering scalability and flexibility. The measurements performed aimed at evaluating the impact of scope partitioning, a participant's speed, the number of participants, and the size of the virtual environment on the platform behavior

4.1. Partition of the Virtual Environment

Partition of the virtual environment into scopes must be accomplished according to the characteristics specified by the application. To verify the impact of this partitioning on the platform behavior, simulations with 32 participants varying the number of groups were performed. The simulation parameters used are those presented in section 3.3.

When there is only one group, the virtual environment has one leader for all participants. Therefore, the platform acts as a centralized system, and the leader is overloaded. There is no data filtering within the virtual world. The world is not partitioned, there is no group changes, and hence there is no interruptions in the perception of the world.

On the other hand, when the number of groups is greater than one, consistency control is distributed among leaders. The larger the number of groups, the fewer participants are controlled by each leader, the greater the distribution of this control, and the smaller the number of update messages exchanged. Nevertheless, there are more interruptions in the virtual environment due to more frequent handoffs.

Figure 4 shows simulations comparing the average delay to enter the virtual world and to perform a handoff. It is important to remark that there is no handoffs when the number of groups is equal to one. The handoff delay is greater than the entrance delay because it is composed by the delay to notify the participant that it must leave the group, and the delay for this participant to enter the virtual world, which is the entrance delay. Therefore, the difference between the two delays stand for the time for the leader to notify the participant that it must perform a handoff.

When the number of groups increases, entrance and handoff delays decrease, inasmuch as the responsibility to send the database to participants is distributed among leaders. When there are fewer participants in each scope, leaders become less overloaded, and can answer join requests faster. This distribution is essential to the scalability of the environment, because when the number of participants increases, the division into groups keeps the processing load approximately constant in each leader. The main element that affects both delays is the time for TCP to transfer the database. Data compression could be used in order to keep this database as small as possible.

Figure 5 shows the total number of handoffs accomplished by all of the participants during a session. It can be noticed that when the number of groups increases, the frequency of handoffs also increases. This can be explained by the fact that an increase in the number of groups corresponds to smaller areas for each group. Thus, for the same probability of movement, this means a larger number of handoffs.

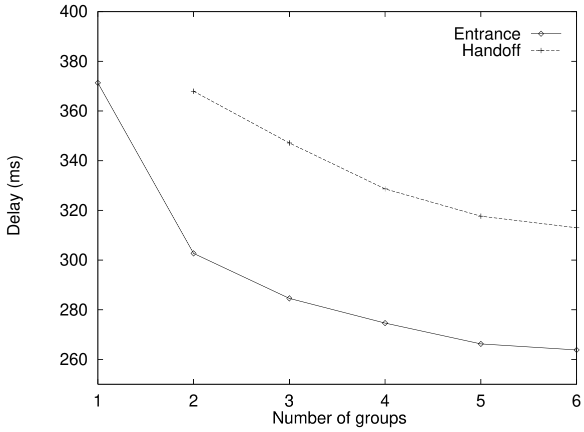


Figure 4. Delay to enter the virtual environment and to perform a handoff.

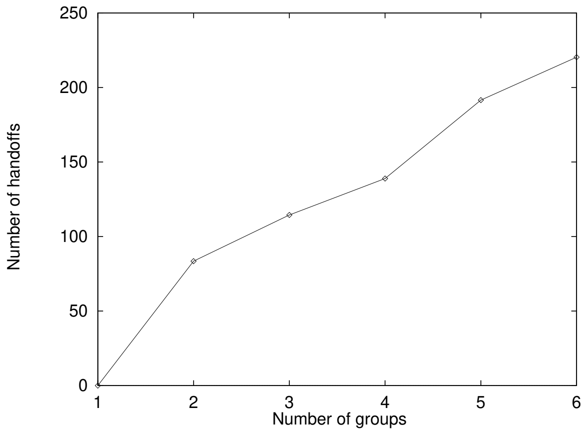


Figure 5. Total number of handoffs for all participants.

Figure 6 shows the number of messages exchanged. It can be observed that the total number of messages exchanged, control messages plus update messages, decreases when the number of groups increases, optimizing the use of communication system resources. By increasing the number of groups interaction scopes are reduced, therefore update messages are delivered to fewer participants. Simultaneously, the frequency of handoffs increases, and hence the number of control messages to perform these handoffs. The reduction of update messages, however, is more significant than the increase in the number of control messages exchanged.

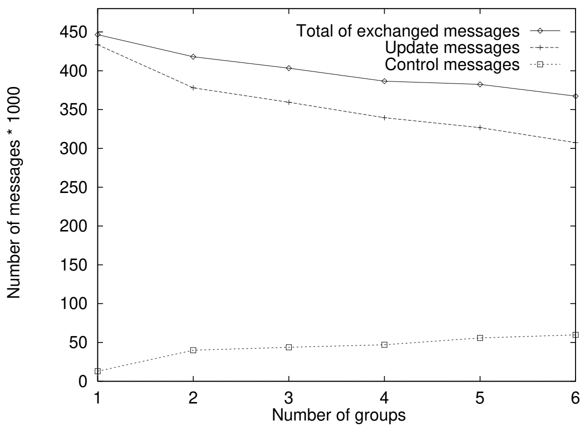


Figure 6. Number of messages exchanged for the topology with 32 participants.

4.2. Participant's Speed

A distributed virtual environment is composed by different types of avatars. On military applications, avatars are ships and airplanes. On games, participants can be represented as ordinary people, flying super-heros, animals, etc. Each one of those avatars has its own characteristics, which determine its behavior in the virtual environment. One of the characteristics that influences most the platform performance is an avatar's speed.

In order to verify the impact of a participant's speed on the platform performance, simulations were performed varying the probability of movement (P_m), parameter that determines a participant's speed. The other parameters remained as presented in section 3.3, and the number of users simulated was 16. Simulations were accomplished both for 4 and for 8 groups.

Figure 7 shows the number of handoffs performed by all of the participants during a session. It can be remarked that the number of handoffs increases linearly when the probability of movement increases. As an avatar's speed increases, participants change groups more frequently, thereby performing more handoffs per session. The number of handoffs is larger when the virtual environment is partitioned into 8 groups than when it is partitioned into 4 groups. For example, when the probability of movement equals 25 %, the number of handoffs grows from 87 for 4 groups to 117 for 8 groups. This is due to the reduction in the area of each group, as discussed in section 3.3.

Nevertheless, the increase in the number of handoffs for higher probabilities of movement has not a significant impact on delays for participants to enter the virtual world nor to perform handoffs, as presented in figure 8. It is important to remark that delays for simulations with 8 groups are smaller than with 4 groups.

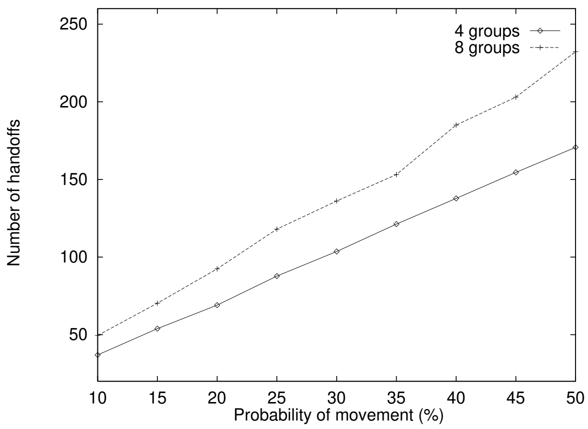


Figure 7. Number of handoffs performed varying a participant's speed.

Consequently, the time of each interruption caused by handoffs is smaller when the environment is more partitioned.

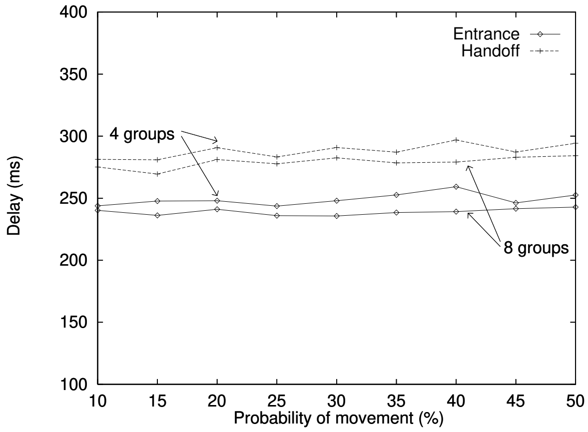
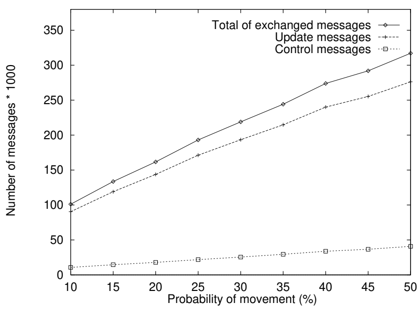
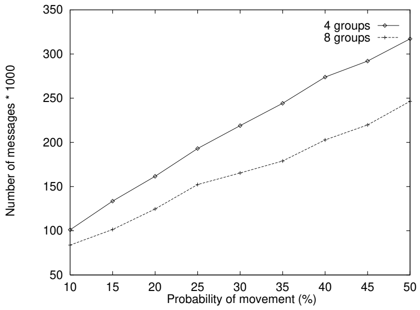


Figure 8. Entrance and handoff delay varying a participant's speed.

Figure 9 shows the impact of varying a participant's speed on network load, i.e., the number of messages exchanged. As the number of handoffs increases, the number of control messages exchanged increases as well. Furthermore, as participants move faster, the number of update messages (ES PDU's) also increases. In figure 9(a), it can be noticed that update messages are the main element in the total number of messages. Therefore, the use of a dead-reckoning algorithm can reduce network load. The division into groups is also important to this reduction, as shown in figure 9(b), that compares the total number of messages exchanged for 4 and 8 multicast groups.



(a) Number of messages with 4 groups.



(b) Comparison between 4 and 8 groups.

Figure 9. Messages exchanged varying a participant's speed.

4.3. Number of Participants

The platform must allow for applications with different number of participants. In order to observe the impact of varying the number of participants on platform performance, simulations were performed for 4 and 8 groups. The parameters used are presented in section 3.3.

Figure 10 shows delay variation when the number of participants increases. Both delays to enter the virtual world and to perform a handoff increase as the number of users increases. In these simulations, the number of leaders is kept constant. Thus, when the number of users increases, the number per leader also increases. Leaders become more overloaded, thereby taking more time to answer join requests and to detect handoffs. It can be noticed, however, that the increase in delay decelerates as the number of users increase. It can also be remarked that the delay is smaller for simulations performed with 8 leaders, allowing a good interactivity with delays lower than 300 ms [17].

The larger the number of users, the larger the number of messages exchanged both to notify a participant's state and to control participants joining and leaving scopes. This increase in the number of messages is shown in figure 11(a). Figure 11(b) presents a comparison between the number of messages exchanged for the simulation with 4 groups and with 8 groups. It can be observed that the simulation with 8 groups reduces the communication-system load, by exchanging fewer messages.

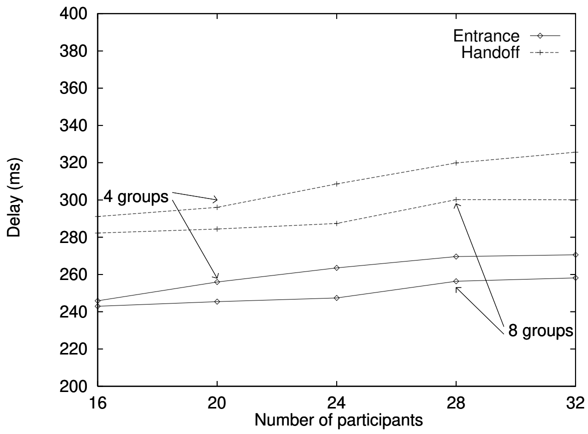
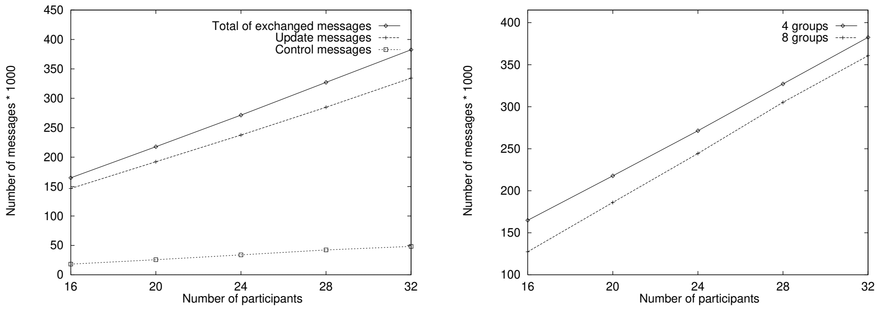


Figure 10. Entrance and handoff delay varying the number of participants.



(a) Number of messages with 4 groups.

(b) Comparison between 4 and 8 groups.

Figure 11. Messages exchanged varying the number of participants.

4.4. Size of the Virtual Environment

The size of the database that represents an interaction scope is one of the elements that determines the delay for a participant to enter the virtual world and to perform a handoff. This size can vary significantly from an application to another depending on the richness of the virtual-world representation.

Simulations to assess platform performance for different scope sizes were performed for 32 participants, dividing the virtual environment into 4 and 8 groups. The other parameters remained as presented in section 3.3. It is important to remark that the parameter size represents just the initial size of a scope. Upon entrance, participants add the size of their avatars to this value. Thus, the size transferred is this initial value plus the size of all of the avatars in a scope.

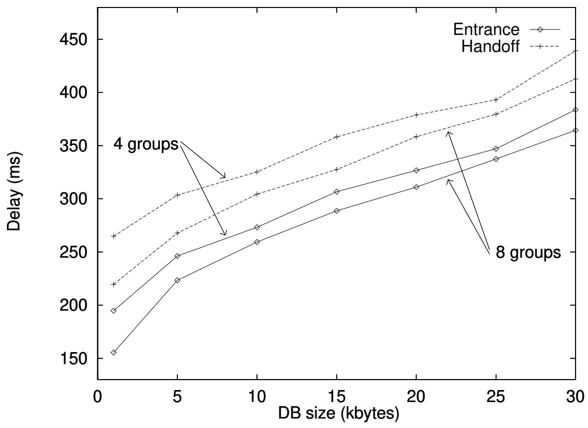


Figure 12. Entrance and handoff delay varying the size of the environment.

Figure 12 presents the delay varying the initial size of the database that represents an interaction scope. It is interesting to notice that there is an inferior bound to both entrance and handoff delays for the topology presented in section 3.2. This minimum delay can be observed when the size of the database is 1 kbyte. For this configuration elements that have more impact on the delay are the time to send a JR PDU and to open a TCP connection to send the JC PDU. It can be remarked that as the size of the database increases, the increase in delay decelerates. This deceleration is due to the segmentation accomplished by TCP, which allows a higher concurrency to transfer the segments of a message.

Figure 13 shows the number of messages exchanged. The number of update messages does not vary significantly, inasmuch as these messages do not depend on the size of the database. On the other hand, the number of control messages increases due to the larger number of segments of JC PDU's, which transfer the database. These messages, however, are not relevant to the total number of messages, which presents a minimal increase.

5. Conclusion

The platform proposed provides a distribution and consistency control service for virtual reality applications aiming at scalability and flexibility. Flexibility was achieved through the specification of a generic protocol, allowing the platform to be used by any distributed virtual reality application. Scalability was achieved using a filtering mechanism along with multicast communication, performing a distributed consistency control accomplished by the leaders, and reducing the exchange of reliable messages.

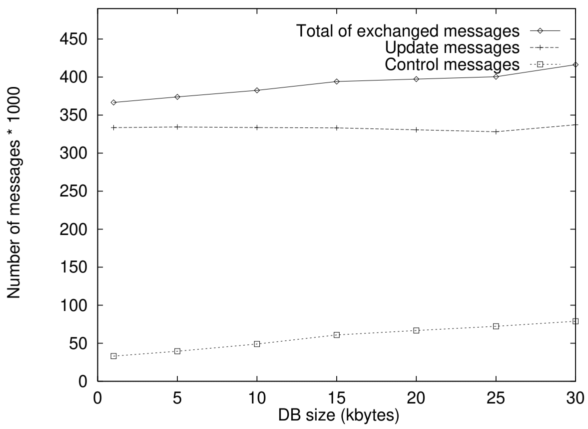


Figure 13. Number of messages exchanged (4 groups) varying the size of the environment.

Simulation results show that the division into groups, by using multicast protocols, causes a reduction in delay both to enter and to change groups, and in the number of messages exchanged. This reduction in delay benefits participants interactivity, and decrease the probability of inconsistencies. There is a tradeoff between this gain in interactivity and the interruptions generated by handoffs when the number of groups increases. Nevertheless, for the simulations performed, interruptions were not significant, standing for less than 1% of the total time of a session for each participant. Division of the virtual environment into groups plays an important role in the system scalability. Despite the increase in control messages, this division reduces the total number of messages and the load in each leader, thereby enabling more participants to share a virtual environment.

For applications with faster avatars, there is a larger number of interruptions due to handoffs and a greater network load. In order to decrease network load, it is interesting to divide more the virtual environment. Considering the worst case, when the probability of movement equals 50%, the number of messages reduces 23.19% when the number of groups increases from 4 to 8. This partitioning, however, increases the number of interruptions of the perception of the virtual environment. When the probability of movement equals 50%, these interruptions stand for 2.69% of the time of a participant session for simulations with 8 groups. For simulations with 4 groups, interruptions stand for 1.37%. Again, time spent on interruptions due to handoffs was not significant. This tradeoff between the number of messages exchanged and the number of interruptions must be adjusted according to the application.

Considering the size of the virtual environment, simulations were performed for a virtual world varying from 4 kbytes (1 kbyte for each scope using 4 groups) to 240 kbytes (30 kbytes for each scope using 8 groups). This size determines the richness of the virtual world. Simulations performed show that the partioning of the virtual environment allows a richer representation of the virtual world, by reducing the delays. Despite this reduction, the size of the virtual world has a great impact on delay and must be minimized in order to avoid large interruptions that would harm interactivity. In the simulations performed dividing the environment into 8 scopes, a good interactivity is achieved for a virtual world of up to 120 kbytes. There are some possibilities to enable the use of larger virtual worlds: compressing the database, or using networks with larger bandwidth, which can reduce the delay to transfer the database.

Results show that platform configuration must be adjusted to the application. Applications with faster avatars can reduce the size of the database in order to decrease the impact caused by interruptions. When it is necessary to express the virtual world with more details, the number of users or a user's speed can be reduced. For all kinds of applications, virtual-world partition have to be performed carefully, optimizing the tradeoff between the size of groups and the number of groups.

In summary, the proposed platform, using a distributed model among leaders and a division of the environment in separated scopes, allowed a good interactivity without harm to perception of the virtual environment.

References

1. Sandeep Bajaj et al. Improving simulation for network research. Technical Report 99-702, University of Southern California, March 1999.
2. John W. Barrus, Richard C. Waters, and David B. Anderson. Locales: Supporting Large Multiuser Virtual Environments. *IEEE Computer Graphics and Applications*, 16(6):50–57, November 1996.
3. S. Deering. *Host Extensions for IP Multicasting.* RFC-1112, August 1989.
4. Michel Diaz. Modeling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models. *Computer Networks and ISDN Systems*, 6(6):419 – 441, December 1982.
5. Kevin Fall and Kannan Varadhan. NS Notes and Documentation. Technical report, The VINT Project, November 1997.
6. Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, 5(6):784 – 803, December 1997.
7. Chris Greenhalgh and Steve Benford. MASSIVE: A Collaborative Virtual Environment for Teleconferencing. *ACM Transactions on Computer-Human Interaction*, 2(3):239 – 261, September 1995.
8. Chris Greenhalgh and Steve Benford. MASSIVE: A Distributed Virtual Reality System Incorporating Spatial Trading. In *Proceedings IEEE DCS'95*, Vancouver, Canada, 1995.
9. Olof Hagsand. Interactive Multiuser VEs in the DIVE System. *IEEE Multimedia*, 3(1):30–39, 1996.
10. Institute for Defense Analysis. SIMNET. May 1990.
11. Michael R. Macedonia. *A Network Software Architecture for Large Scale Virtual Environments.* PhD thesis, Naval Postgraduate School, Monterey, California, June 1995.
12. Michael R. Macedonia and Michael J. Zyda. A Taxonomy for Networked Virtual Environment. *IEEE Multimedia*, 4(1), 1997.
13. Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Paul T. Barham, and Steven Zeswitz. NPSNET: A Network Software Architecture for Large Scale Virtual Environments. *Presence: Teleoperators and Virtual Environments*, 3(4):265 – 287, 1994.

14. Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Donald P. Brutzman, and Paul T. Barham. Exploiting Reality with Multicast Groups. *IEEE Computer Graphics and Applications*, pages 38 – 50, September 1995.
15. Ken Pimentel and Brian Blau. Teaching Your System to Share. *IEEE Computer Graphics and Applications*, 14(1), January 1994.
16. Sridhar Pingali, Don Towsley, and Jim Kurose. A Comparison of Sender-Initiated and Receiver-Iniciated Reliable Multicast Protocols. *IEEE Journal on Selected Areas in Communications*, 15(3):398 – 406, April 1997.
17. Standard Commitee on Interactive Simulation, IEEE Computer Society. IEEE Standard for Distributed Interactive Simulation. IEEE Std 1278.1-1995, 1995.
18. Martin R. Stytz. Distributed Virtual Environments. *IEEE Computer Graphics and Applications*, 16(3):19 – 31, May 1996.
19. E. W. Zegura, K. Calvert, and S. Bhattacharjee. A Quantitative Comparision of Graph-Based Models for Internet Topology. *IEEE/ACM Transactions on Networking*, 5(6):770 – 783, December 1997.