Machine Intelligence & Robotic Control, Vol. 1, No. 1, 11–26 (1999)

Paper

Application of a Multi-User Distributed Virtual Environment Framework to Mobile Robot Teleoperation over the Internet

Maja Matijašević[†], Kimon P. Valavanis[‡], Denis Gračanin[§], and Ignac Lovrek[†]

Abstract: This paper presents a framework for multi-user distributed virtual environments (VEs) and explores its application to teleoperation of a mobile robot over the Internet. The proposed framework, incorporating the functional and the interconnection models, attempts to represent common functionality, communication issues, and requirements found in distributed multi-user VEs. A distributed virtual environment for mobile robot teleoperation has been implemented based on distributed simulation and virtual reality standards and used for measurements in the laboratory testbed. It has been demonstrated that user behavior and interactions need be taken into account when specifying networking requirements for distributed VEs.

Keywords: Virtual Reality, Mobile Robot, Teleoperation, Modeling, Distributed Simulation

1. Introduction

VIRTUAL reality (VR) is becoming increasingly recognized as a technology offering numerous benefits to (tele)robotics [7]. In addition to being safer, less tedious, and less costly than real task performance testing, VR provides an intuitive, experiential human interface and provides visualization of simulated actual, as well as hypothetical situations, while incorporating real constraints. Manufacturing and teleoperation, particularly teleoperation with inadequate visual feedback, as well as supervisory and collaborative control, are only some areas where the "synergy between VR and robotics" [11] is expected to grow. On the other hand, VR in combination with high-speed networking technologies provides a basis for distributed virtual environments (VEs). Such VEs allow multiple remote users, as well as simulated or real entities (e.g., robots!), to participate and interact in shared virtual worlds.

This paper explores how a (rather general) distributed VE framework is applied to mobile robot teleoperation over the Internet. As such, the purpose of this paper is twofold: A framework for distributed multi-user VEs is first discussed that attempts to represent common functionality, communication issues, and requirements found in such VEs. This framework tends to be general and suitable for a variety of applications. Then, teleoperation using a VR interface (virtual environment), whether in a single-user or a multi-user mode of operation, is demonstrated as a highly interactive distributed application. For such applications, adequate networking support in terms of delay and throughput is necessary to meet "real-time" constraints in terms of human perception.

* Received August 23, 1999; accepted September 26, 1999.

† University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Telecommunications, Unska 3, HR-10 000 Zagreb, Croatia. E-mail: {maja, lovrek}@tel.fer.hr

‡ University of Louisiana at Lafayette, Center for Advanced Computer Studies, Lafayette, LA 70504-4330, USA. Also with the Technical University of Crete, Dept. of Production Engineering and Management, Chania, Greece. E-mail: kimon@cacs.usl.edu

§ University of Louisiana at Lafayette, Virtual Reality and Multimedia Laboratory, PO Box 44932, Lafayette, LA 70504-4932, USA. E-mail: dg@acim.usl.edu

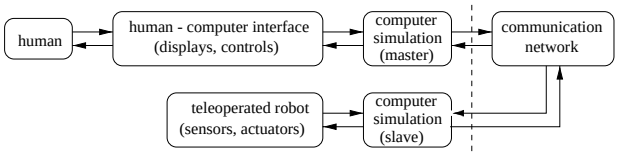

Fig. 1 Outline of teleoperation/telerobotic VE

In further detail, for mobile robot teleoperation over the Internet, the teleoperation/telerobotic VE shown in **Fig. 1** [16] is followed. An interprocess communication is established to allow the human to operate a virtual robot (master) in the VE, where this operation is in turn reflected on the actual robot (slave) in "real-time." This basic outline may be extended to create a distributed multi-user VE, such that multiple users may view the ongoing experiment. However, the robot teleoperation must be exclusive, i.e., only one user at a time may operate the robot, although users may alternate in the role of the robot operator. Similarly, this concept may be applied to multiple robots and multiple operators. When this is the case, each user may control his/her own mobile robot, viewing simultaneously through the VE interface the operation of other robots. However, issues of mobile robot navigation coordination and collision avoidance are robotics issues and are not a part of the VE. A prototype application for mobile robot teleoperation has been developed and used for measurement of networking requirements using several single user and multiple users scenarios in the laboratory testbed. It is essential to emphasize that the main benefits of such a multi-user system are the ability to train (remote) mobile robot operator(s) by allowing them to see and to perform mobile robot (tele)operation, as well as to rehearse and execute collaborative mobile robot scenarios.

Although at this time Internet provides only a "best-effort" (no guarantees on delay/delay variation) transport service, the Next Generation Internet/Internet 2 will provide quality of service (QoS) support. While QoS is an open area of research, it is clear that requirements for a particular application need be specified. For distributed VEs, the problem of specification of networking requirements

is complicated by the fact that they are often application specific and usage scenario dependent. Thus, models of distributed VEs found in literature are generally geared towards application at hand, as opposed to the more general framework presented here.

In summary, the paper contributions are:

- Derivation of a framework for distributed VEs.
- Application of the framework to mobile robot teleoperation over the Internet.
- Development and testing of a prototype multi-user distributed VE for mobile robot teleoperation in a laboratory testbed.
- Experimental results that strengthen the hypothesis that user behavior and interactions need be taken into account when specifying networking requirements for such distributed VEs.

The rest of the paper is organized as follows: Section 2 briefly presents related work. Section 3 summarizes the framework for distributed multi-user VEs, incorporating the functional and the interconnection model. Section 4 describes the experimental study, a prototype application for mobile robot teleoperation, its implementation and testbed. Section 5 presents and discusses the measurements' results. Section 6 concludes the paper.

2. Related Work

A detailed review of related research and work in the field of distributed multimedia and virtual reality applications for the framework described in Section 3 may be found in [26]. Reviewed distributed VEs include NPSNet [25], Virtual Prototyping System [24], DIVE [22], mWorld [15], MASSIVE-2 [21], and Spline [4].

Teleoperation over the Internet and the World Wide Web [9], [13], [19] is gaining popularity in applications ranging from art to telemedicine [18], [31], [32]. Since the pioneer *Mercury Project* [19] in 1995, more than twenty robots are currently publicly accessible on the Web (http://ranier.oact.hq.nasa.gov/telerobotics_page/realrobots.html).

The Internet and the Web have been further enhanced with 3D graphical visualization and interactivity through the Virtual Reality Modeling Language (VRML) [2]. In general, VRML provides a means for creating 3D objects and worlds, sensing and specifying responses to user interaction, key frame animation, scripting, and prototyping. In addition to shaded and textured 3D geometry, VRML-modeled virtual worlds may be enhanced by background, panorama, lights, sound, video, and animation. Several examples of use of VRML in teleoperation over the Web have been reported [3], [23], [27].

3. Distributed VE Framework Fundamentals

In a distributed multi-user VE application, two relevant questions are: (i) *How does an interaction at the application (user) level affect the communication characteristics*, and, (ii) *How are events at the communication level reflected at the application level*.

These questions have been already studied [26] from the simplest case of a single user local (non-distributed) VE to the most general and complicated case of multiple users distributed VE. Two derived models, the functional and the interconnection models, comprising the overall framework, provide an answer to these questions. The models represent two complementary, interrelated views of a distributed VE, as shown in **Fig. 2**. Using components of the VE system [16], a distributed VE may be viewed as a collection of (multiple) user processes and autonomous simulation processes that communicate over the network. This communication is depicted as ***Network I/O*** block. The functional model hides the distributed nature of the overall VE beyond the ***Network I/O***. Complementary to the functional model, the interconnection model addresses the distributed view, and expands on the ***Network I/O***, without distinction between user and autonomous simulation processes.

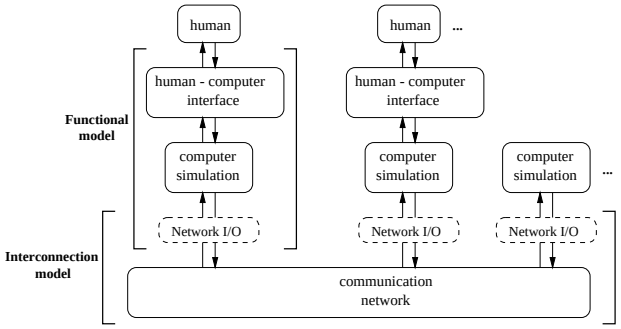

Fig. 2 The concept of two complementary models

The models are briefly described below, using the Unified Modeling Language (UML) [8]. For the UML specification, the reader is referred to http://www.rational.com/uml/. UML defines several types of diagrams for graphical notation. *Statechart diagrams* and their variation, *activity diagrams*, are used in the representation of the functional model. Statechart diagrams are used to describe the behavior of the system. Activity diagrams focus on activities representing the operations and sequencing of related activities. In the interconnection model, the UML *class diagrams* are used for static structure description, and the *interaction diagrams* (including *sequence diagrams* and *collaboration diagrams*) are applied for the description of interactions between UML components.

3.1 Functional model

The functional model for multi-user distributed VE is shown in **Fig. 3**. It consists of four groups of functions that represent the core of a VE, namely, ***User input***, ***Compute***, ***Display***, and ***Network I/O***. The user is included in the loop between the ***Display*** and ***User input***.

To accommodate different (multi-)user interactions the ***User input*** is divided in two components: ***User-VE*** (that includes navigation, user–object, and user–user interactions), and ***User–group*** (that includes membership and participation interactions). The ***Display*** includes ***VE display*** and ***Non-VE display***, where the former displays the virtual world, and the latter status and error notifications.

Following the grouping of input and display functions, it is convenient to separate the functions in ***Compute*** into ***Virtual world management*** and ***Session management***. The ***Virtual world management*** deals primarily with users' actions and their effects within the VE and manages the dis-

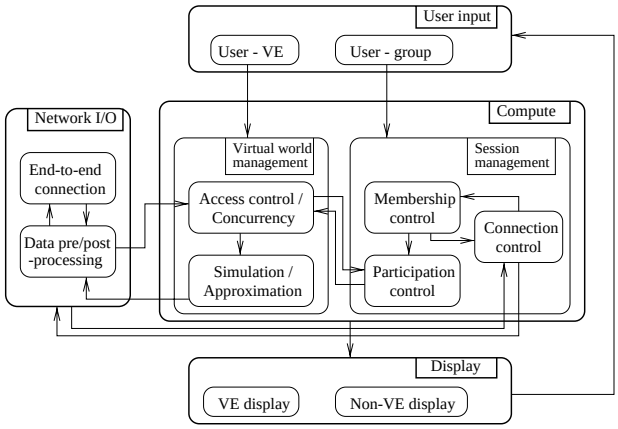

Fig. 3 Functional model for multi-user distributed VE

tributed data, while the ***Session management*** focuses on users' membership, participation, and communication between distributed processes that constitute the VE.

The ***Virtual world management*** is responsible for the virtual world content. The virtual world is composed of virtual *objects* placed in a common VE. An object is characterized by a list of *attributes* and *behaviors*. The selection of attributes and behaviors that adequately characterize the object depends on the object itself, as well as on the use of attributes and behaviors in the specific application. Attributes that describe the properties of an object are referred to as *proper attributes*, and attributes that describe object's relation to the environment are referred to as *spatial attributes*. For example, an object may have proper attributes that describe its shape (e.g., sphere), appearance (e.g., shiny, red color), and physics (e.g., mass 1 kg), and spatial attributes that describe its position and orientation. Behaviors may be divided into two broad categories as deterministic and non-deterministic. Behaviors triggered by user actions are typically non-deterministic. A formal description of behaviors (not to mention standardization) is still an open research issue. Attributes and behaviors of an object may be changed by the simulation, based on passage of simulation time and processing of the user input.

Simulation, in general, may be any combination of a ***Visual simulation***, an ***Audio simulation***, a ***Physical simulation***, and an ***Application-specific simulation***. (The ***Application-specific simulation*** is not considered further as a part of the general model.) ***Approximation*** is tightly coupled with the ***Simulation*** and is therefore represented in the same block. In a distributed VE, the control over virtual world entities may be distributed among users and simulations in order to reduce the local processing at each site. The purpose of ***Approximation*** is to approximate the behavior of remote entities based on the updates received from the controlling ***Simulation***. An approximation is typically much less computationally extensive than the simulation. On the sending side, an approximation is run in parallel to simulation, and the results are compared in order to decide whether an update is needed for remote entities. For remote entities, only the object audio-visual properties need be known, and the scene is updated based on updates received over the network and the approximation algorithm (e.g., *dead reckoning* [1]).

Access control/Concurrency implements the control of the local user's actions and concurrency restrictions due to multi-user access, serialization, and synchronization of multiple inputs, necessary for preserving consistency. The ***Access control/Concurrency*** operates in conjunction with ***Participation***, and it implements either access matrix based rules, or a role-based policy. The use of role-based policies may assist in limiting some of the concurrency problems, as demonstrated by their application in collaborative multi-user applications [17]. The ***Participation*** deals with any user-to-role assignment.

Concurrency control is implemented on top of access control, with the goal of preserving integrity of shared virtual world. Usual approaches for concurrency control for shared data include locking and transactions at the data level. A higher level of control may be implemented based on analysis of user interactions. The interaction-related concurrency may be restricted or resolved by assigning and negotiating ownership (per object/attribute) and/or roles (per user). This leads to a more general basis for access rules.

The ***Session management*** has two aspects in distributed multi-user applications: one related to *social aspect*, i.e., meeting of a *group* of users in a shared VE, and the other addressing *communication aspect*.

The social aspect of session management determines how users join and leave (i.e., membership), and the policy regarding users' interactions (i.e., participation). Session management in terms of communication separates the control needed during the data transfer from the data transfer itself [33]. It includes different groups of tasks, related to session administration, session configuration, establishment/release, and control of synchronized exchange of information between peers. Two dominant session models for multi-user applications are the light-weight session and tightly-coupled session. They differ in how the issues of membership, participation, and connection establishment are addressed.

Group control comprises ***Membership control*** and ***Participation control***. The ***Membership control*** enables a user to become a session member (join the shared application), or end the membership (leave the shared application). The ***Membership control*** and ***Connection control*** are related through granting a membership (a "join" request) which may imply a session establishment, or renegotiation of session (quality of service, QoS) parameters, whereas issuing a "leave" request may imply the current session release. In addition, the ***Membership control*** provides a link to the ***Participation control*** to allow the user to set initial set of access rights or role within the VE. The association of users with roles and implementation of a participation policy is the main purpose of ***Participation control***. The ***Participation control*** may also allow dynamic roles, where the role initially assigned to user may be renegotiated.

The ***Network I/O*** includes ***Data pre/post-processing***, and ***End-to-end connection***. The ***Data pre/post-processing*** represents non-context-specific handling of different types of discrete data (such as position updates) and continuous data (such as audio streams). Context-specific processing, such as functional/logical partitioning [25], [30], is handled within ***Simulation/Approximation***, since only the

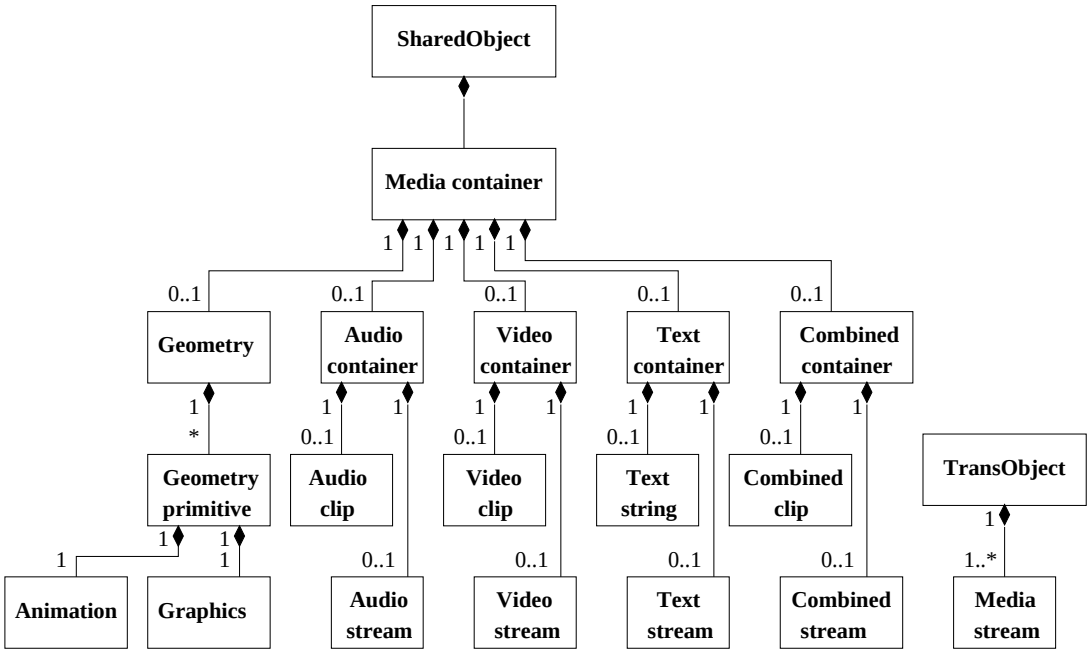

Fig. 4 SharedObject and TransObject decomposition

simulation "knows" about the context. This enables appropriate media-specific methods, including compression and media-specific filtering (e.g., hierarchical encoding), to be applied for reducing the network load, and also allowing the users to choose what media types, and with what quality, they wish to receive. On the sending side, the ***Data pre/post-processing*** acts as pre-processing, with the output written to the ***End-to-end connection***. On the receiving side, the ***Data pre/post-processing*** does the post-processing, i.e., reconstruction to original form.

The ***End-to-end connection*** represents an abstraction of a transport protocol. Its functionality is to establish end-to-end transport-level connection(s), enable exchange of various types of information while maintaining satisfactory performance, and close the transport connection(s). The ***End-to-end connection*** may also perform mapping of partitioned data/streams to multicast groups. The different requirements on the network in terms of distribution, timing, reliability, and other performance requirements are determined by QoS parameters. The mapping of application requirements to transport QoS parameters is performed by the session service, and it is handled by the session management. The choice of an appropriate set of QoS parameters may be not only a media-specific, but also an application-specific issue. Proposed parameters include large scale multicast applications [6], and parameters for media streaming applications [5], [28], [33].

3. 2 Interconnection model

The interconnection model represents a distribution-oriented view of a VE. Three aspects addressed include distribution of users, distribution of data, and distribution of processing.

Distribution of users is an intrinsic property of a multi-user VE. It partially influences the distribution of data and processing since at least a portion of data, as well as local VE input and display processing, are usually local to the user's computer. In addition, the number of users and topology of their distribution determine the user co-located multiple endpoints of communication. The session model for multi-user distributed VEs is based on the concept of *group (or multipoint) communication* [14]. A group corresponds to a set of participating processes, where processes include, but are not limited to, user processes. Three components may be identified as necessary for a group system support [12]: group management, group communication, and replication management. Group management is needed to create and destroy a process group, and to (optionally) keep record of group membership that changes through join and leave requests. Group communication represents a mechanism for information exchange among group members, while the goal of replication management is to maintain consistency between local data copies associated with multiple processes within a group. The responsibility for maintaining consistency is given to application processes (that decide on when and how the updates are sent, as well as what they contain), and the communication infrastructure that transfers the updates over the network. One of the main concerns of a distributed application is to maintain consistency, while allowing multiple user interactions. Due to non-deterministic processes (different user inputs), an exchange of information is necessary to achieve consistent replication.

The analysis of VE content and the discussion of group communication in the interconnection model highlight the different requirements for *replication of shared objects* vs. *streaming of continuous media objects* at the transport level. The VE transport requirements extend the fundamental requirements for continuous media by involving application aspects, such as (i) the manner media are composed within VE objects, and (ii) user's perception of, and interaction with, VE objects.

A classification of objects is proposed that divides objects into *SharedObjects* and *TransObjects*. A SharedOb-

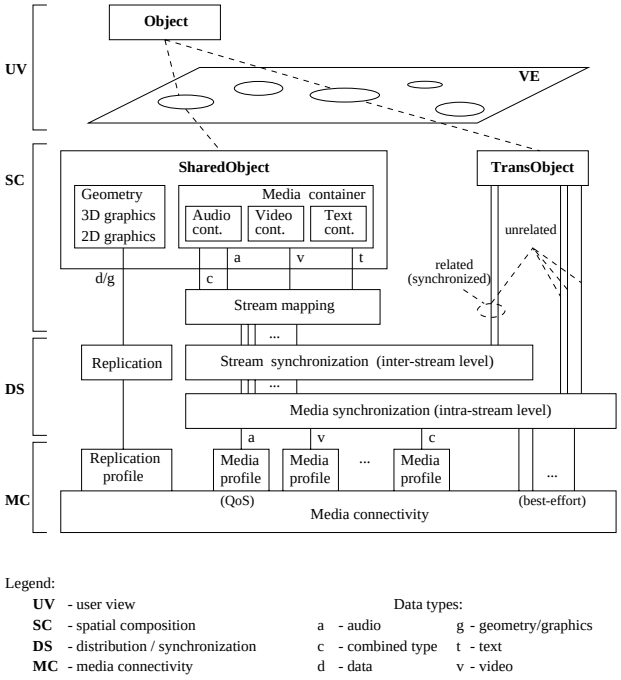

Fig. 5 The interconnection model

ject, shown in **Fig. 4**, contains geometry, and a media container for each data type. Data types include single media types (audio, video, and text) and a combined media type. A combined type represents a unified presentation of multiple media, such as audio/video, audio/text, audio/video/text, etc. The purpose of media containers is to define spatial attributes of media elements, for example, the position of a sound source in space. Two data types are distinguished for each medium, based on replication requirements: (i) stored media (clip), and (ii) real-time (live feed) streaming media. Stored media replication is reduced to one-time (download or streaming) transport for each clip, while for live media there is a transport-layer virtual circuit established for communication between the source and the sink for the whole duration of the session. The terminology for shared objects is adopted from the draft specification of Web 3D Consortium (http://www.vrml.org) *Living Worlds* working group and it is extended with terminology for continuous media objects. A *TransObject* (Fig. 4) is an object containing continuous media data, e.g., streaming video, audio, or text. Such an object is not replicated: instead, a streaming connection between the source and sink is established. Media streams in TransObject do not have a spatial component, and are presented to the user via non-VE display. A TransObject provides only minimal control interface to the user, allowing the user to stop unwanted stream(s), as well as temporarily pause the stream delivery (*play*, *pause*, and *stop*).

Figure 5 illustrates the interconnection model. The model shows mapping of virtual world objects from the user view (UV) to media connectivity (MC). The UV corresponds to an individual user's experience of the common VE. In the user view, a VE is a collection of virtual world objects, including the environment, static and dynamic objects, and embodiments of other users or processes. In this view, the distribution of the virtual environment is transparent.

The next two views represent different levels of composition and decomposition within the application. First is the spatial composition (SC). SharedObjects are presented using a VE display, and TransObjects are presented using a non-VE display. Media components in SharedObjects differ from those in TransObjects in their spatial component, so for shared objects, the next view is the spatial composition. At this level, a SharedObject is represented by geometry (data that requires reliable transport and replication) and a number of media containers (audio, video, text, combined). A media container represents a spatial "place holder" for a media stream. Different types of media are mapped to their respective containers. Information about replicated objects is contained within their spatial attributes, and does not require any additional processing. An example for spatial composition is avatar's voice appearing as "bound" to the movement of the avatar.

The distribution and synchronization (DS) level focuses on temporal interdependency. For shared objects, there is a temporal relationship between replicas on distributed machines. In the model, this is shown as replication associated with SharedObjects. A TransObject contains references to multiple media streams, which may be either unrelated or temporally related in a common presentation. Stream synchronization is needed for all such streams, regardless of whether they are mapped to media containers in a SharedObject, or presented directly to the user as a TransObject. Before mutual dependency is addressed, media synchronization is applied for each single media stream separately. This may require less processing if the stream encoding already contains synchronization information (combined type). The mutual dependency between streams may be determined easily if they belong to the same source, for example, captured sound in motion video, or if the streams are mapped to media containers of the same shared object. However, whether there is a dependency between streams via their respective objects can not be determined by the model.

The media connectivity (MC) level deals with mapping of application level QoS parameters to transport. For example, replicated data transport may be mapped to reliable multicast. Different media streams have different requirements on reliability, bandwidth, and timing (delay and jitter), which is also dependent on sampling, encoding, and compression. For example, both audio and video may tolerate some loss, but they may not tolerate jitter. Combined types may be less susceptible to jitter, since they are restored at the receiver based on built-in synchronization information. The overall requirements for different media types may be grouped and matched to parameters and options of existing standards. This mapping may be defined as a *profile* for a given media type. Media streams may be mapped to profiles that describe media requirements in terms of QoS for the transport layer. In addition to media profiles defined per media type, it may also be possible to negotiate QoS on per-connection basis.

4. The Experimental Study

The prototype multi-user networked VE provides teleoperation of a mobile robot (Nomad 200, Nomadic Tech-

Table 1 VRML virtual world model components

VRML nodes	Object/Option
Geometry	Floor, walls (Blue Room)
Appearance	Obstacle unit
	Virtual robot
	Avatar (not visualized)
Environment	Light
	Background
Navigation	Walk
	Teleport
	Examine
Viewpoint	Top view
	Side view(s)
	Corner view(s)
	Robot camera view

nologies, Inc.) in the Robotics and Automation Laboratory (RAL) of the University of Louisiana at Lafayette. The Virtual Reality and Multimedia Laboratory has been used for software development. The prototype application has been developed based on the following requirements:

- Create a multi-user virtual environment, where the virtual world represents the robot in the so-called *Blue Room* (BR), the actual set-up within RAL.
- Integrate the mobile robot in the virtual environment and enable teleoperation of the robot.
- The control over the robot has to be exclusive (i.e., only one user at a time may control the robot); however, multiple users may view the ongoing experiment.
- A virtual control device (*control panel*, CP) is used to control the robot. The user who controls the robot (temporarily) owns this device. An extension of basic functionality includes a transfer of control between users.

The VE has been implemented using a combination of standard components: VRML, IEEE 1278 Distributed Interactive Simulation (DIS), and Java. By complying to these standards, rather than be limited to specific hardware, a virtual world could potentially be viewed on any computer with a network connection and a Web browser. Thus, users may be physically in different locations, and view and participate in an experiment over the Internet. However, due to unpredictable delays in wide-area networks, real-time control over the best-effort (no bounds on delay!) Internet may be limited to users in a local area network.

4.1 VRML

In the prototype VE, VRML has been used to model the virtual world, representing the mobile robot and its environment, the so called *Blue Room*, with the components as in **Table 1**.

Human avatars have not been modeled in VRML, but each viewer (browsing user) is treated as an autonomous entity whose position and orientation are updated. The real robot has been integrated in the VE, visualized as a virtual robot in the VE, and a real-time user control (teleoperation) of the robot has been enabled.

In a multi-user VE, only one user at a time may operate the robot. Multiple users may view the ongoing experiment, and alternate in the role of the robot operator. The robot is controlled via a virtual control panel (CP), with components summarized in **Table 2**. User may also navigate in the virtual world, using the browser's standard navigation dashboard. A snapshot of the user interface with the virtual environment and the CP upon initialization and after acquiring the operator privilege is shown in **Fig. 6**. **Figure 7** shows the standard Cognos 2D interface that only provides a 2D map of the environment. A series of snapshots of the user interface after the robot is moved are shown in **Figs. 8** and **9**. The VR interface, on the other hand, allows viewing from predefined points including top view (top) and corner view (bottom), as well as unconstrained viewing as shown in Fig. 8 (top, bottom).

Table 2 Virtual robot control panel components

Group of commands	Command
Movement controls	Forward, Backward
	Turn left/right
	Stop, Zero
Joystick control	Forward, Backward
	Turn left, Turn right
	Stop when released
Control transfer	Get robot, Release robot
	Grant control, Deny control
Status displays	Operator status (true/false)
	Robot status (free/busy)

4.2 Java implementation of DIS

DIS standard [1] is based on a concept of a synthetic environment consisting of computationally autonomous simulators (simulation software applications) running on geographically distributed, networked host computers. DIS defines a set of 27 PDUs for communicating events between autonomous simulators. In practice, a simulation may use any subset of DIS PDUs.

In this application, Java implementation of DIS (by DIS-Java-VRML working group [10] of the Web 3D Consortium) has been used. Further information on DIS-Java-VRML may be found at http://www.stl.nps.navy.mil/dis-java-vrml.

4.3 Distributed VE design

The final design is shown in **Fig. 10**. Three control components include: the *VE initialization control*, the *robot client-mode control*, and the *robot direct-mode control*. In the robot direct-mode control, client communicates direct with the robot deamon. In the robot-client mode control, the real robot (robot deamon) receives commands through the robot simulator, using a one-to-one TCP/IP connection. The robot simulator (*Nomad simulator*) is a part of the Cognos Host Software Development Environment, from Nomadic Technologies, Inc. The version 2.6.7. used in the testbed includes an API and software libraries written in C programming language. A separate stand-alone Java application has been developed to communicate and control the mobile robot using a wireless Ethernet link.

The VE initialization uses a WWW browser (HTTP over TCP/IP) for one-time download of the initial virtual world and the virtual control device that substitute the Nomad

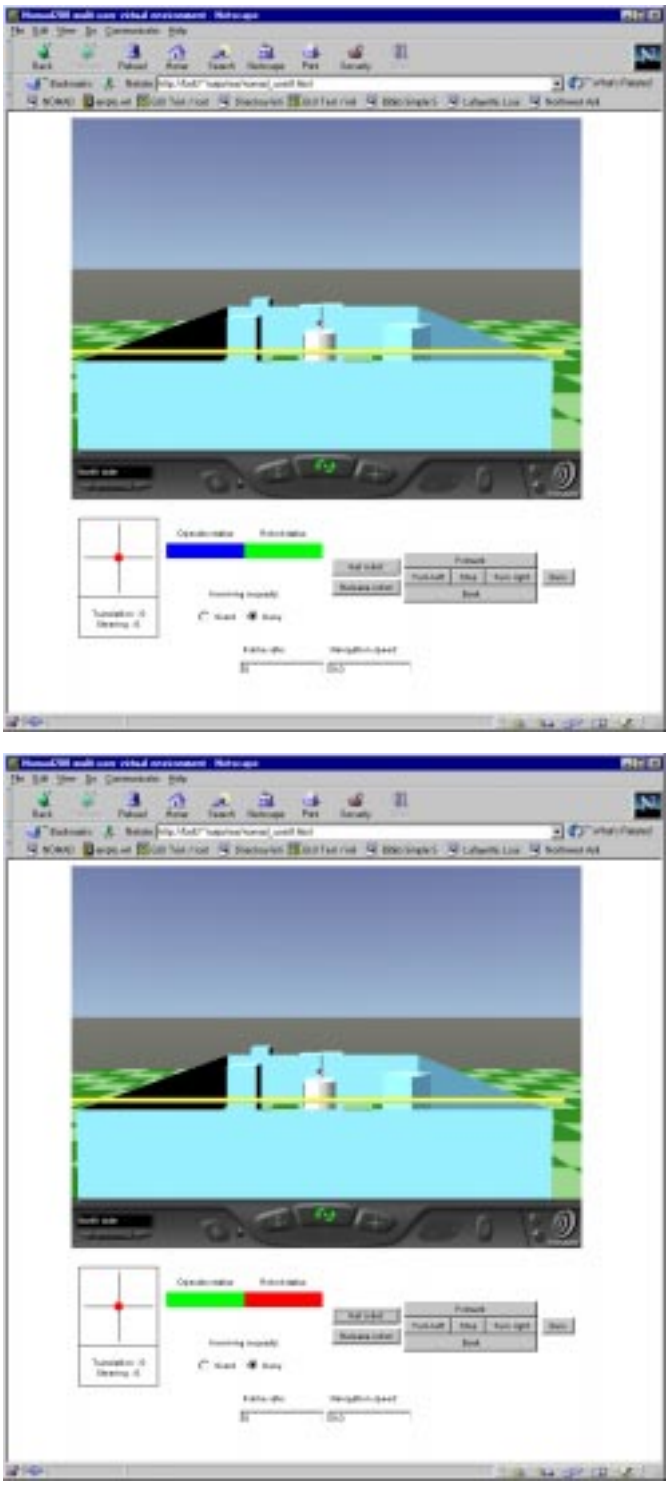

Fig. 6 VR user interface upon initialization (up) and after becoming an operator (down)

GUI. The initial virtual world is displayed in the browser's VR-plugin window, and the virtual control device is a Java applet embedded in the same WWW page. The communication between the Java applet and the VRML browser uses the External Authoring Interface (EAI). For details on EAI, the reader is referred to the Web3D Consortium. To establish communication with the Nomad server from the virtual world, the Java applet also implements the Nomad client, using the locally developed implementation of Nomad API in Java [20].

The multi-user VE uses DIS over IP multicast (UDP/IP). To use DIS for the communication between the Nomad client and the Nomad server, both client and server are extended with *DIS interpreters* implemented in Java. On the client side, the DIS interpreter encapsulates the Nomad API control parameters within DIS PDUs before multicasting them over the network. On the server side (robot client mode control), the DIS interpreter reads DIS PDUs from the network, extracts Nomad server commands, and writes them to the server (TCP) communication port. The server executes the command and responds by multicasting the updated state of the robot. DIS PDUs are also used to im-

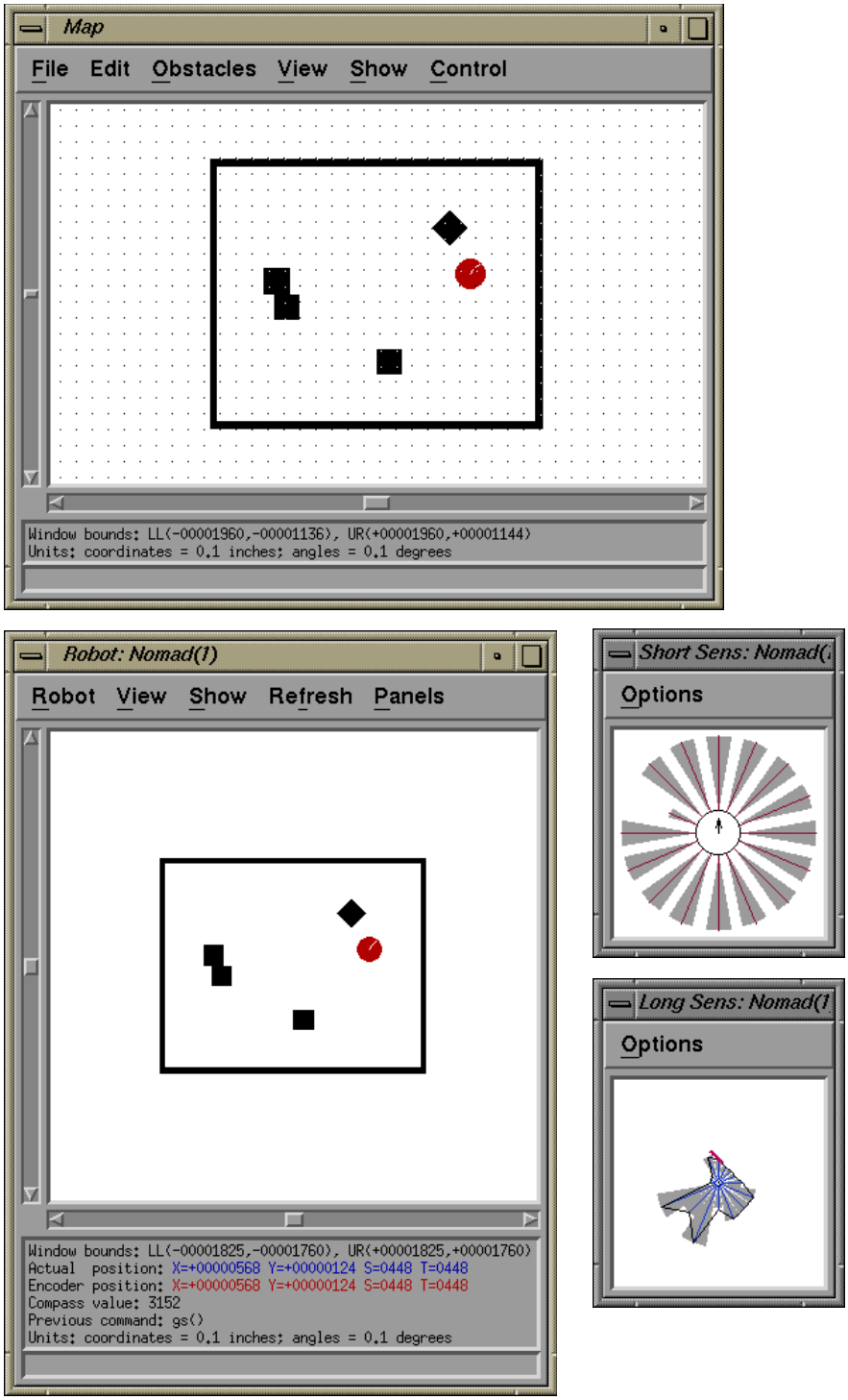

Fig. 7 A snapshot of the Nomad's 2D user interface

plement exclusive control of the robot in a multi-user environment.

4. 4 The functional and the interconnection model

The functional model for the case study corresponds to the one in Fig. 3. The ***User input*** and ***Display*** are integrated within the browser, as well as the visual simulation within the ***Virtual world management***. (The mobile robot physical simulation is an autonomous process.) One of the application's requirements is that only one user at a time can control the robot. The means to control the robot is the CP. For the purpose of this application, two user roles are defined as *spectator* and *(tele)operator*. By default, users join the VE as spectators. The user who controls the robot temporarily assumes the role of the operator. To assure that at any one time, there may be only one operator, the ***Access control/concurrency*** is centralized and co-located with the robot simulator, while the ***Participation control*** controls a user's role in the VE. The negotiation and transfer of operator privileges is handled through message passing and events pertaining to access (current operator and robot status) are visualized using color indicators on the CP. The

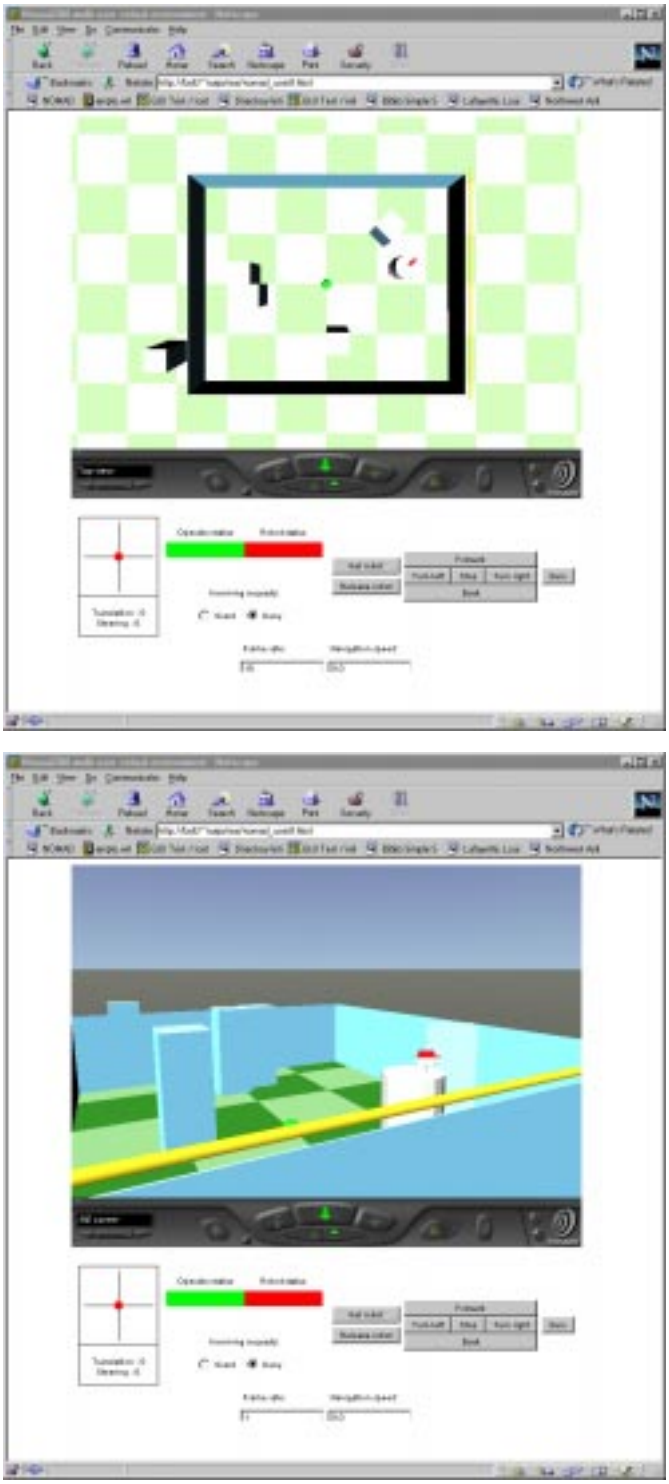

Fig. 8 Some predefined views of the VE: top view (up) and corner view (down)

Membership control handles how users join and leave the VE. A light-weight session model is adopted, which does not require an explicit session setup. Users may join and leave at any time. Timeouts are implemented to avoid deadlock situations, for example, when current operator leaves the VE without releasing the CP. Message passing, described in more detail later, is handled by the ***Network I/O***.

The interconnection model for the case study may be simplified as shown in **Fig. 11**, since all dynamic objects in the scene are modeled as SharedObjects, and there are no TransObjects. More advanced applications may require Transobjects for incorporating real-time sensor data from the robot in the VE.

In the model, the ***User view*** is represented by a user interface, i.e. the browser. ***Spatial composition*** describes the mapping between objects in the user view and replicated data. This functionality is implemented behind the user interface, through software (Java applet) that controls the VR display, and communicates with the network, as described earlier in this Section. At the ***Distribution/synchronization*** level, ***Replication*** uses the DIS protocol to exchange mes-

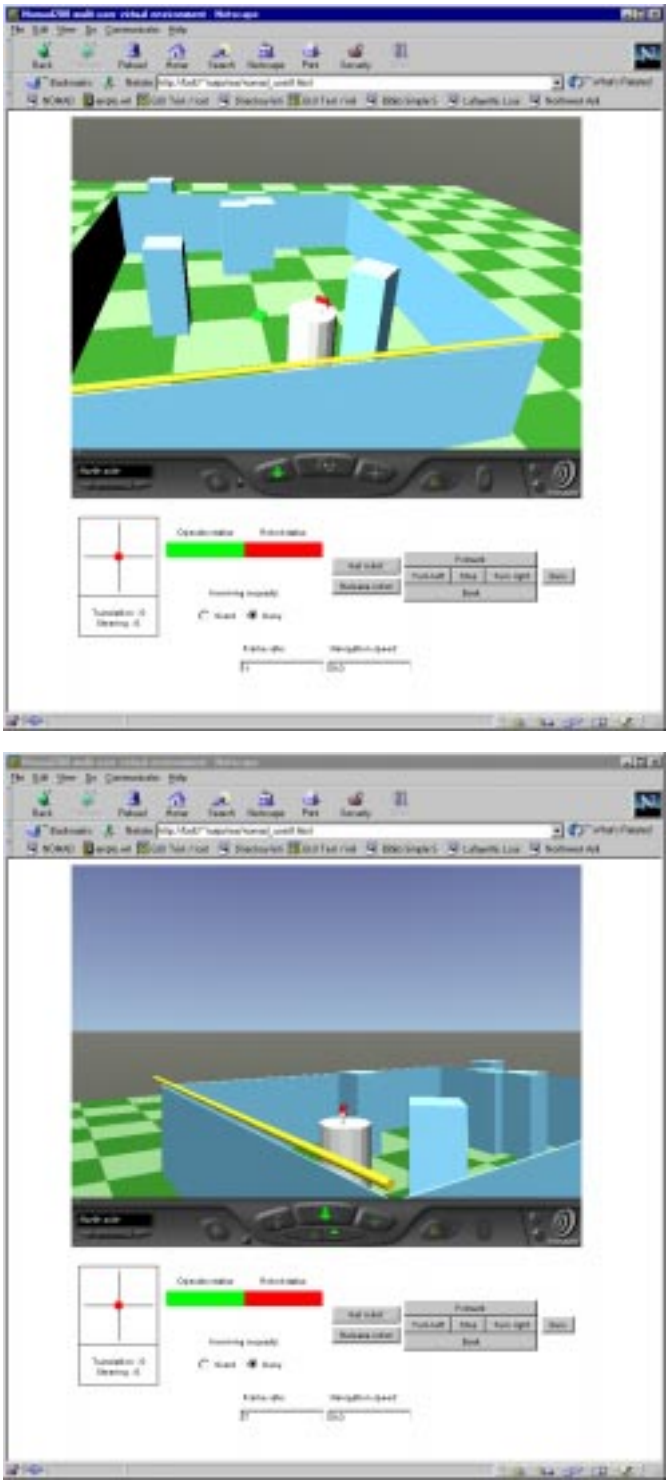

Fig. 9 Unconstrained views of the VE

sages for purposes of replication and access control. A subset of DIS PDUs for this VE includes the Entity State PDU (ESPDU), Action Request PDU (AREQ), and Action Response PDU (ARES). The ESPDUs are used for replication, i.e., communicating the updated state of the virtual world objects, including position, orientation, and appearance. The AREQ, and ARES PDUs are used for negotiation and transfer of access control to robot, as well as for teleoperation of the robot. The ***Replication profile*** at the ***Media connectivity*** level represents a combination of protocols used for communication over the network, i.e., the DIS running over UDP/IP multicast.

5. Experimental Results and Discussion

Experimental results in several single user and multiple users scenarios using a laboratory testbed are presented next.

5. 1 Testbed description

A heterogeneous computing environment consisting of the following platforms has been used as a testbed:

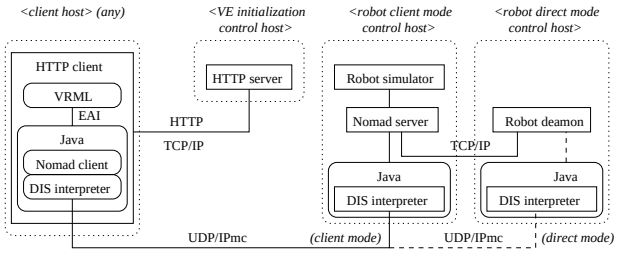

Fig. 10 Design of a multiple user distributed VE

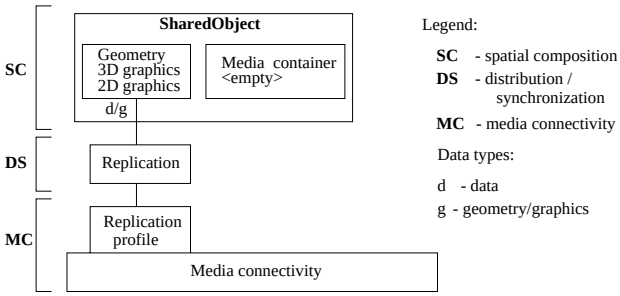

Fig. 11 Interconnection model for the case study

- A Nomad 200 mobile robot base (from Nomadic Technologies Inc.) equipped with a shared memory multiprocessor control system. The master processor is a Pentium-based PC (Pentium 133 MHz, 32 MB RAM), running Red Hat Linux release 3.0.3 (Picasso), Kernel 2.0.24;
- Several Silicon Graphics workstations, (Indy, Indigo, Onyx, and O_2), running IRIX 6.5.4;
- Several Gateway 2000 PCs (Pentium 200 MHz/ 128MB RAM and Pentium II 333 MHz/384 MB RAM), running Microsoft Windows 95 and Windows NT.

All computers used in a multi-user configuration are connected to a 10 Mb/s Ethernet LAN. The robot uses a wireless Ethernet link to RangeLAN2/Access Point (from Proxim, Inc.) to communicate with other computers in a LAN. Distributed VE has been implemented as shown in Fig. 10, with SGI O_2 as the VE initialization control host and robot client mode control host, and other SGI workstations and PCs as client hosts. The robot direct model control host is Nomad 200. Measurements were performed in several single and multiple user scenarios.

5.2 Single user scenarios

In single user scenarios, illustrating typical user interactions related to robot operation and navigation, PDU statistics have been collected. These measurements give the typical PDU throughput and patterns for different ways of controlling the robot.

5.2.1 Scenario S1 (Robot control using buttons) In this scenario, a user starts the application and waits for the indication of the robot status. Initially, the robot is free and the user acquires control over the robot, thus becoming an operator. The task to perform in the VE is to move the robot using buttons on the CP, following an imagined rectangular path around the Blue room.

Table 3 PDU statistics

Scenario	Entity State	Action Request	Action Response	Total
S1	297	29	29	355
	83.66%	8.17%	8.17%	100%
S2	152	216	216	584
	26.02%	36.99%	36.99%	100%

5.2.2 Scenario S2 (Robot control using soft joystick) As in the previous scenario, the user starts the application, waits for the indication of the robot status, and initially, acquires control over the robot, thus becoming an operator. The user performs the same task as in Scenario S1, but using exclusively the virtual joystick to move the robot.

The PDU statistics are summarized in **Table 3**.

In both scenarios, the majority of PDUs are ESPDUs, which is usual for DIS applications [29]. The number of Action Request PDUs is equal to the number of Action Response PDUs (i.e., no loss of PDUs has been observed). The amount of Action Request and Action Response PDUs depends on user interactions and the implementation of the virtual controls. Initial transfer of control from robot to user requires one request/response pair for each successful exchange of control. The overhead related to the exchange of control is thus predictable. The manipulation of the robot results in a variable number of PDUs, depending on the performed task.

In Scenarios S1 and S2 the user accomplishes the same task, however, in Scenario S1 with the use of buttons, fewer PDUs are generated than in Scenario S2 with the use of joystick. The reason for this lies in the implementation of virtual controls. The motion commands using buttons result in one PDU request/response being exchanged, and the issued command is valid until the next command is received or a timeout occurs. Joystick, on the other hand, generates commands whenever its position changes, and it resets the motion parameters to zero when released. With quantization of joystick updates, 10–20 request/response pairs in one "burst" are generated for maximal virtual joystick deflection.

5.2.3 Scenario S3 (User navigation modes) In Scenario 3, the effect of an operator's and a spectator's behavior has been studied. Most VRML browsers support several types of navigation, including walk, teleport, and examine. **Figures 12–16** illustrate the throughput for different navigation types. All PDUs are ESPDUs.

It may be noted that the teleportation style of navigation generates more PDUs as the user's viewpoint rapidly changes. Walking through the VE generates (in average) less traffic than in case of teleportation, however, walking periods are clearly distinguishable from the "quiet state" in Scenario M1 (presented next).

5.3 Multi-user scenarios

The scenarios presented next demonstrate the use of the application with ten and sixteen simultaneous users, and illustrate dependency of network requirements on the number and the behavior of users. Throughput, delay, and PDU statistics have been measured. Due to multicast and software compatibility limitations in previous versions used

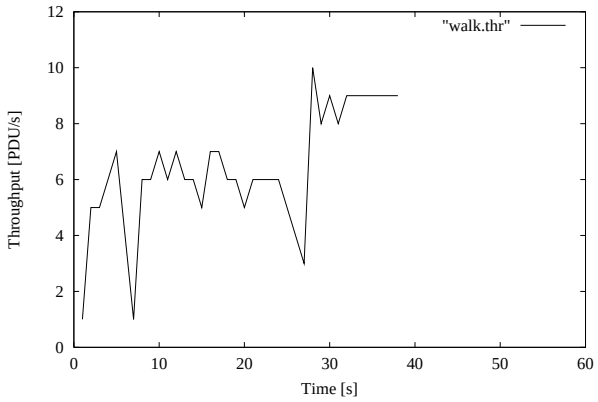

Fig. 12 Throughput in Scenario S3 (walk)

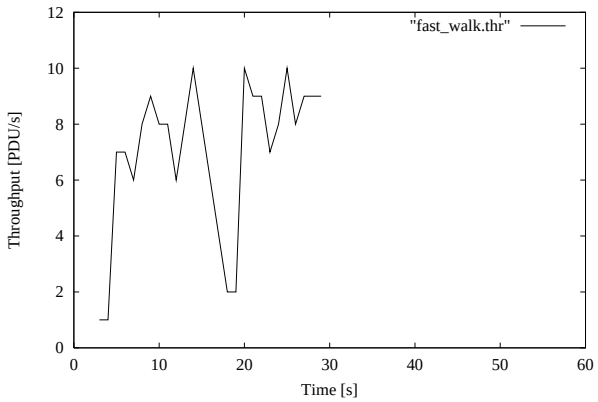

Fig. 13 Throughput in Scenario S3 (fast walk)

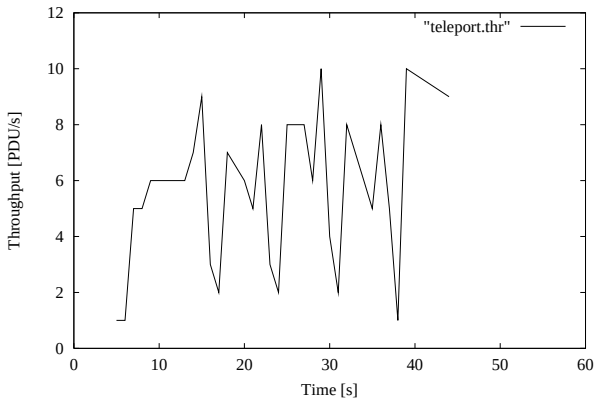

Fig. 14 Throughput in Scenario S3 (teleport)

in the testbed, scenarios M1–M4 include ten simultaneous users, and scenarios M5–M7 include sixteen simultaneous users.

The components from Fig. 10 were set as follows: VE initialization control host: SGI O_2; client hosts: 10 users (10 PCs); 16 users (10 PCs and 6 workstations); robot client mode control host: SGI O_2; and robot direct model control host: Nomad 200.

The snoop packet capture program, included as a part of the IRIX operating system distribution, has been used for data collection. The collected data has been analyzed using different filters to extract the packets of interest and examine the relevant packet fields (time-stamps, DIS ES-

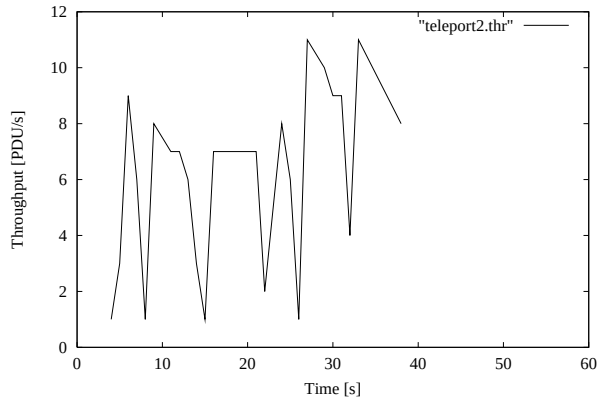

Fig. 15 Throughput in Scenario S3 (teleport)

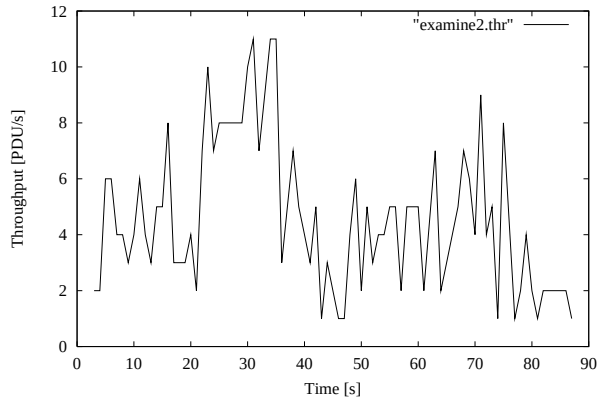

Fig. 16 Throughput in Scenario S3 (examine)

PDU header fields, etc.).

5.3.1 Scenario M1 (Bandwidth requirements for inactive users) For the purpose of this scenario, after initialization of the robot, users join the VE at an (approximate) rate of 1 user/minute. Both the robot and users remain inactive for the duration of the experiment. Thus, this scenario addresses the "quiet state" where the traffic only consists of keep-alive messages. **Figure 17** shows the multicast throughput for controlled join of ten users, and **Fig. 18** for sixteen users. The throughput is shown in relation to the number of users (PDU sources) including the robot. All PDUs recorded in this scenario are Entity State PDUs, representing keep-alive messages, periodically generated by users and the robot.

5.3.2 Scenario M2 (User navigation styles and bandwidth requirements) Scenario M2 starts with ten users present in the VE. Independently of each other, the users navigate through the VE using the browser built-in commands (browser dashboard) and predefined viewpoints. **Figure 19** shows the overall multicast throughput. All PDUs recorded in this scenario are ESPDUs since no exchange of control occurs. Navigation is characterized by "bursts" of ESPDUs denoting the change of user's viewpoint position and/or orientation.

5.3.3 Scenario M3 (Unsuccessful exchange of control) In Scenario M3, ten users are initially present in the VE. One user acquires the control over the robot, thus becoming an operator. The default response to incoming

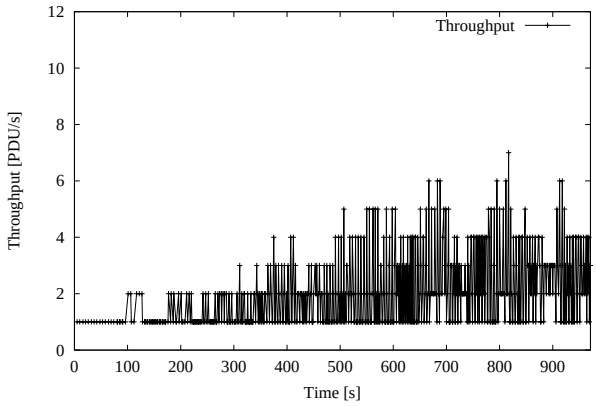

Fig. 17 Multicast throughput in Scenario M1 (ten users)

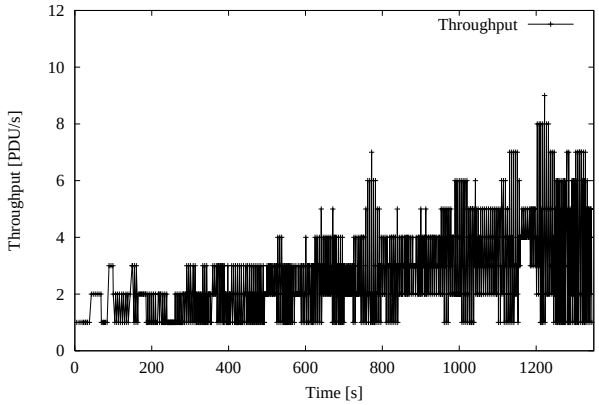

Fig. 18 Multicast throughput in Scenario M1 (sixteen users)

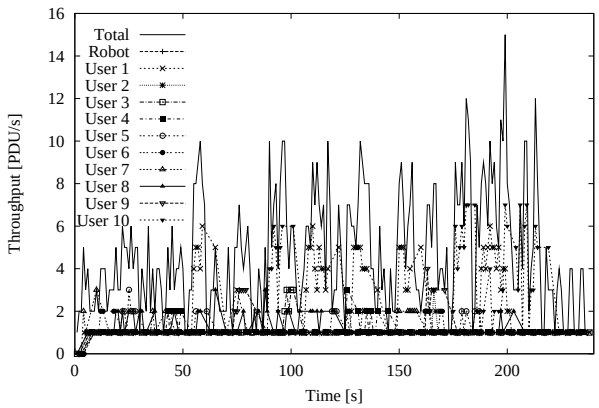

Fig. 19 Multicast throughput in Scenario M2

requests for exchange of control is set to *Deny*. Independently from each other, other users then attempt to gain control over the robot by sending requests to the current operator. The requests are denied and no transfer of control occurs. **Figure 20** shows the multicast throughput and delay between Action Request and Action Response PDUs. It may be noted that the delay results are scattered in two regions centered around the borders of the simulation interval. PDU statistics shows that majority of PDUs are Entity State PDUs (77.22%), and the rest are Action Request/Action Response (11.39%, 11.39%).

Table 4 PDU statistics for M3 and M4

Scenario	Entity State	Action Request	Action Response	Total
M3	454	67	67	588
	77.22%	11.39%	11.39%	100%
M4	1116	252	252	1620
	68.8%	15.6%	15.6%	100%

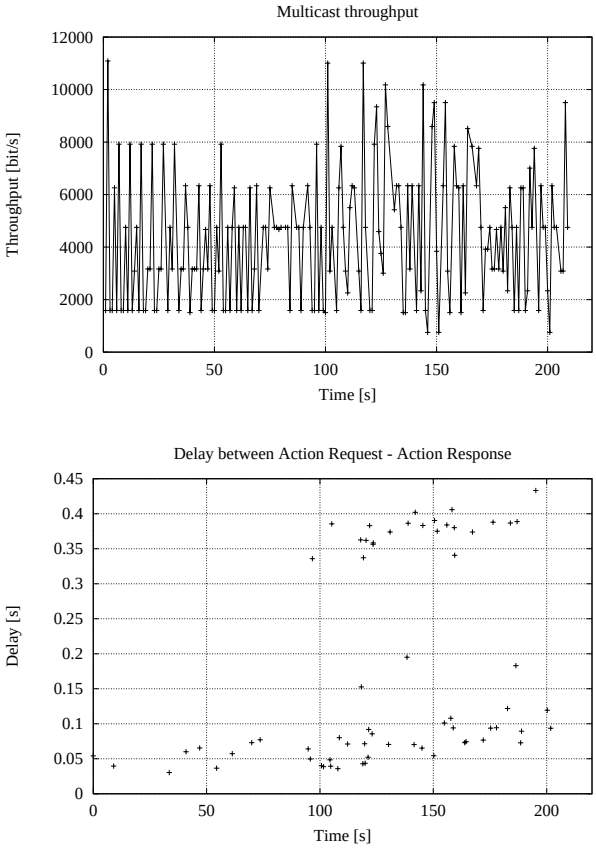

Fig. 20 Multicast throughput and delay in Scenario M3

5.3.4 Scenario M4 (Successful exchange of control) In Scenario M4, ten users are initially present in the VE. One user acquires the control over the robot and sets the default response to incoming requests for exchange of control is set to *Grant*. Other users also change this setting to *Grant* which enables every request to be granted and the exchange of control to occur every time a request is issued. **Figure 21** shows the multicast throughput and delay between Action Request and Action Response PDUs. PDU statistics for scenarios M3 and M4 are given in **Table 4**.

PDU statistics shows Entity State PDUs (68.8%) and a relative increase in Action Request/Action Response PDUs (15.6%, 15.6%). Compared to Scenario M3, the average throughput is higher and average delay longer in Scenario M4. The successful exchange of control typically generates a sequence of 4 PDUs: the Action Request/Action Response pair, and 2 ESPDUs, one from the previous operator and one from the new one, denoting a change of ownership, while the unsuccessful exchange of control only involves the Action Request/Action Response pair. The successful exchange of control in Scenario M4 and more user activity are also responsible for longer av-

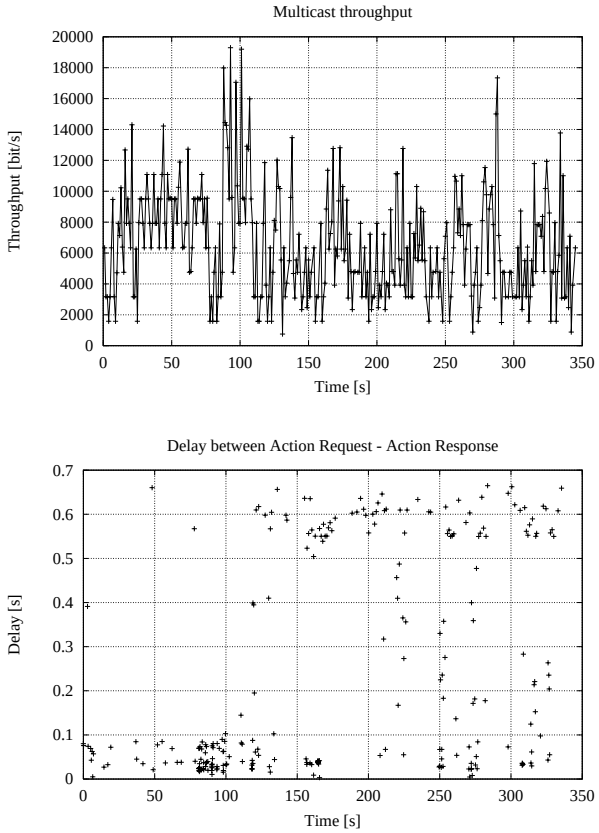

Fig. 21 Multicast throughput and delay in Scenario M4

erage delay, due to additional PDU processing as well as potentially mis-directed requests. The latter may happen when the exchange of control happens quickly several times in a row, so the requests sent to the previous operator before an identity of the new operator becomes known through multicast update.

5.3.5 Scenario M5 (Multiple activities and button teleoperation) Scenarios M5 and M6 study the "realistic" throughput with sixteen simultaneous users. These scenarios are realistic in the sense that they are less restrictive regarding user's behavior, and thus give an idea of throughput when application is used without restrictions on users' behavior.

The difference is in teleoperation activity: in Scenario M5, teleoperation is performed using buttons, and in Scenario M6, it is performed using the joystick. Knowing the details of the user interface described earlier, scenarios M5 and M6 represent the best and the worst case scenario (regarding traffic caused by teleoperation), respectively. All types of navigation are allowed for the operator, as well as the spectators. Transfer of control is not prescribed either: the operator may choose whether to grant or deny incoming requests.

In Scenario M5, sixteen users are initially present in the VE. User navigation, as well as transfer of control is happens randomly, but the robot operator only uses buttons for robot teleoperation. This scenario may be considered as lower bound of throughput during active participation. **Figure 22** shows the multicast throughput in Scenario 5. Maximum throughput in this scenario is 22,176 bit/s, and average throughput is 6,974.12 bit/s. PDU statistics in Scenarios M5 and M6 are given in **Table 5**.

Table 5 PDU statistics for M5 and M6

Scenario	Entity State	Action Request	Action Response	Total
M5	1248	64	65	1376
	90.69%	4.65%	4.65%	100%
M6	1827	311	311	2449
	74.60%	11.88%	11.88%	100%

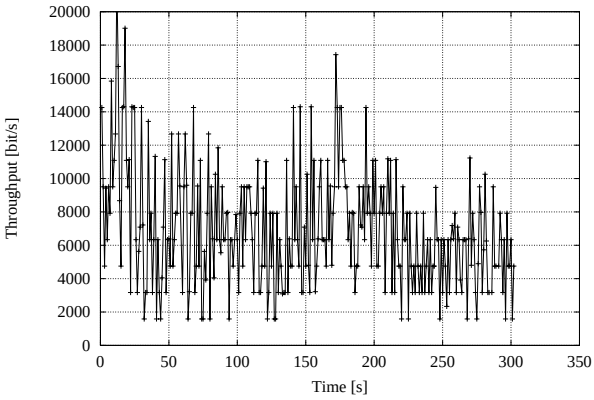

Fig. 22 Multicast throughput in Scenario M5

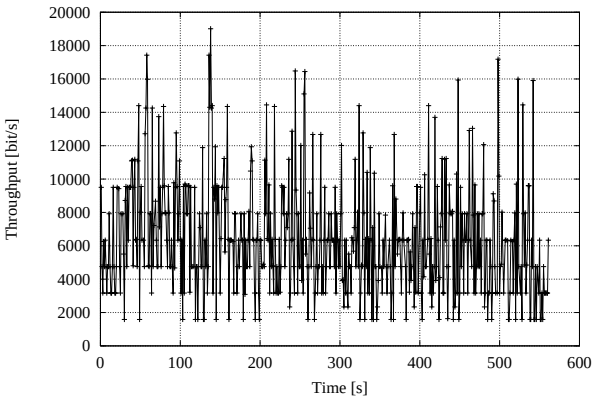

Fig. 23 Multicast throughput in Scenario M6

5.3.6 Scenario M6 (Multiple activities and joystick teleoperation) In Scenario M6, sixteen users are initially present in the VE. User navigation, as well as transfer of control is happens randomly, but the robot operator only uses the soft joystick for robot teleoperation. **Figure 23** shows the multicast throughput for Scenario 6. Maximum throughput in this scenario is 19,008 bit/s, and average throughput is 6,362.13 bit/s.

6. Conclusions

A framework for multi-user distributed virtual environments (VEs) has been presented. Application of the framework to teleoperation of a mobile robot over the Internet has been demonstrated, using a prototype multi-user distributed VE, implemented based on distributed simulation and virtual reality standards. The proposed framework, incorporating the functional and the interconnection models, and description of the virtual world objects are not applica-

tion or implementation specific, and they provide a solid basis for description of a networked VE. Measurements conducted in several single-user and multi-user scenarios demonstrate that the networking requirements for interactive distributed VE depend not only on media types but also on user behavior and interactions. Applicability of the proposed framework to mobile robot teleoperation has been demonstrated and evaluated experimentally.

Acknowledgments

This research has been partially funded by the National Science Foundation under Grants BES-9506771 and BES-9712565, and Louisiana Board of Regents Support Fund Grant 98-00-ENH-TR-92R.

References

[1] *IEEE Standard for Distributed Interactive Simulation*. New York, NY: The Institute of Electrical and Electronics Engineers, Inc., Apr. 5 1996, IEEE Std 1278-1995.

[2] *Information Technology — Computer graphics and image processing — The Virtual Reality Modeling Language (VRML) — Part 1: Functional specification and UTF-8 encoding, ISO/IEC 14772-1*. International Standards Organization, 1997.

[3] J. Alex, B. Vikramaditya, and B. J. Nelson, "Teleoperated micromanipulation within a VRML environment using Java," in *Proc. of the 1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 1998, vol. 3, pp. 1747–1752.

[4] D. B. Anderson, J. W. Barrus, J. H. Howard, C. Rich, C. Shen, and R. C. Waters, "Building multi-user interactive multimedia environments at MERL," *IEEE Multimedia*, vol. 2, no. 4, pp. 77–82, Winter 1995.

[5] R. T. Apteker, J. A. Fisher, V. S. Kisimov, and H. Neishlos, "Video acceptability and frame rate," *IEEE Multimedia*, vol. 2, no. 3, pp. 32–40, Fall 1995.

[6] P. Bagnall, R. Briscoe, and A. Poppitt, Taxonomy of communication requirements for large-scale multicast applications. IETF Large Scale Multicast Applications Working Group, Internet draft, May 1999, work in progress.

[7] A. K. Bejczy. "Virtual reality in robotics," in *Proc. of the 1996 IEEE Conf. on Emerging Technologies and Factory Automation*, 1996, pp. 7–15.

[8] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley, 1999.

[9] K. Brady and T.-J. Tarn, "Internet-based remote teleoperation," in *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, May 1998, vol. 1, pp. 65–70.

[10] D. Brutzman, "The Virtual Reality Modeling Language and Java," *Communications of the ACM*, vol. 41, no. 6, pp. 57–64, 1998.

[11] G. C. Burdea, "The synergy between virtual reality and robotics," *IEEE Trans. on Robotics and Automation*, vol. 15, no. 3, pp. 400–410, June 1999.

[12] F. J. N. Cosquer and P. Veríssimo, "Survey of selected groupware applications and platforms," ESPRIT Basic Research Project 6360: BROADCAST Technical report 40, ESPRIT, 1994.

[13] B. Dalton and K. Taylor, "Distributed robotics over the Internet," *IEEE Robotics and Automation Magazine*, vol. 7, no. 1, Mar. 2000, to be published.

[14] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint communication: A survey of protocols, functions, and mechanisms," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 277–290, Apr. 1997.

[15] J. M. S. Dias, R. Galli, A. C. Almeida, C. A. C. Bello, and J. M. Rebordão, "mWorld: A multiuser 3D virtual environment," *IEEE Computer Graphics and Applications*, vol.17, no. 2, pp. 55–65, Mar.–Apr. 1997.

[16] N. I. Durlach and A. S. Mavor, Eds., *Virtual Reality: Scientific and Technological Challenges*. Washington, D.C., WA: National Academy Press, 1995.

[17] W. K. Edwards, "Policies and roles in collaborative applications," in *Proc. of the ACM Conf. on Computer supported cooperative work*, Boston, MA, Oct. 22-26 1996, pp. 11–20.

[18] H. Friz, P. Elzer, B. Dalton, and K. Taylor, "Augmented reality in Internet telerobotics using multiple monoscopic views," in *Proc. of the 1998 Int. Conf. on Systems, Man, and Cybernetics*, Oct. 1998, pp. 354–359.

[19] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, "Desktop teleoperation via the World Wide Web," in *Proc. of the 1995 IEEE Int. Conf. on Robotics and Automation*, May 1995, vol. 1, pp. 654–659.

[20] D. Gračanin, M. Matijašević, A. Chandrupatla, P. Subramaniam, and S. L. V. Ippagunta, "Integration of mobile robot into virtual reality testbed," in *Proc. of the 1998 IEEE Int. Conf. on Control Applications*, Trieste, Italy, Sept. 1998, vol. 2, pp. 1302–1306.

[21] C. Greenhalgh and S. Benford, "A multicast network architecture for large scale collaborative virtual environments," in *Multimedia Applications, Services and Techniques — ECMAST '97, Proc. of the Second European Conference*, Milan, Italy, May 21–23 1997, vol. 1242 of *LNCS*, pp. 113–128.

[22] O. Hagsand, "Interactive multiuser VEs in the DIVE system," *IEEE Multimedia*, vol. 3, no. 1, pp. 30–39, Spring 1996.

[23] H. Hirukawa, T. Matsui, S. Hiraki, K. Konaka, and S. Kawamura, "A prototype of standard teleoperation systems on an enhanced VRML," in *Proc. of the 1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems IROS'97*, Sept. 1997, vol. 3, pp. 1801–1806.

[24] V. D. Lehner and T. A. DeFanti, "Distributed virtual reality: Supporting remote collaboration in vehicle design," *IEEE Computer Graphics and Applications*, vol. 17, no. 2, pp. 13–17, Mar.–Apr. 1997.

[25] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, "NPSNET: A network software architecture for large scale virtual environments," *Presence: Teleoperators and Virtual Environments*, vol. 3, no. 4, pp. 265–287, Fall 1994.

[26] M. Matijašević, *Distributed Networked Multimedia and Virtual Reality Applications for Multi-User Environment*. PhD thesis, University of Zagreb, Zagreb, Croatia, Dec. 1998.

[27] O. Michel, P. Saucy, and F. Mondada, "'KhepOnTheWeb': An experimental demonstrator in telerobotics and virtual reality," in *Proc. of the Int. Conf. on Virtual Systems and Multimedia VSMM'97*, Sept. 1997, pp. 90–98.

[28] C. Nicolau, "An architecture for real-time multimedia communication systems," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 391–400, Apr. 1990.

[29] J. M. Pullen and D. C. Wood, "Networking technology and DIS," *Proceedings of the IEEE*, vol. 83, no. 8, pp. 1156–1167, Aug. 1995.

[30] B. Roehle, "Channeling the data flood," *IEEE Spectrum*, vol. 34, no. 3, pp. 32–38, Mar. 1997.

[31] R. Simmons, J. Fernandez, R. Goodwin, S. Koening, and J. O'Sullivan, "Xavier: An autonomous mobile robot on the Web," *IEEE Robotics and Automation Magazine*, vol. 7, no. 1, Mar. 2000, to be published.

[32] M. R. Stein, "Painting on the World Wide Web: The Puma Paint project," *IEEE Robotics and Automation Magazine*, vol. 7, no. 1, Mar. 2000, to be published.

[33] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*. Upper Saddle River, NJ: Prentice-Hall PTR, 1995.

Biographies

Maja Matijašević received her Dipl.-Ing. (1990), M.Sc. (1994), and Ph.D. (1998) degrees in Electrical Engineering from the University of Zagreb, Croatia, and the M.Sc. in Computer Science (1997) from the University of Louisiana at Lafayette (former University of Southwestern Louisiana), Lafayette, LA, USA. Since 1991 she has been affiliated with the Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, where she currently holds a research and lecturing assistant position. In 1996–1998 she was a research associate at the Center for Advanced Computer Studies of the University of Louisiana at Lafayette, LA, where she completed most of her Ph.D. thesis work. Her main research interests include computer and telecommunication networks, multimedia, and virtual reality. Ms. Matijašević is a member of IEEE and Upsilon Pi Epsilon.

Kimon P. Valavanis received the Diploma in Electrical Engineering from the National Technical University of Athens, Athens, Greece, in 1981, and the M.Sc. and Ph.D. degrees in Electrical Engineering, Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1984 and 1986, respectively. Since 1991 he has been with The Center for Advanced Computer Studies, University of Louisiana at Lafayette (former University of Southwestern Louisiana), where he is a professor of computer engineering and the director of the Robotics and Automation Laboratory. He is also the A-CIM/BORSF Regents Professor of Manufacturing. His research interests are in the areas of robotics, automation, and distributed intelligence systems. Currently (academic year 1999-2000) he is with the Technical University of Crete, Greece, Department of Production Engineering and Management. He has published more than 180 technical papers, book chapters, and technical reports. He has been the general chair and the program chair of IEEE conferences and symposia, and an editor of several conference proceedings. He is also an Editor-in-Chief of the IEEE Robotics and Automation Magazine. Dr. Valavanis is a Senior Member of IEEE.

Denis Gračanin received the B.Sc. and M.Sc. degrees in Electrical Engineering from the University of Zagreb, Croatia, in 1985 and 1988, respectively. He also received the M.Sc. and Ph.D. degrees in Computer Science from the University of Louisiana at Lafayette (former University of Southwestern Louisiana), LA, in 1992 and 1994, respectively. Between 1985 and 1991 he was a lecturer assistant at the Telecommunications Department of the University of Zagreb. He is currently a research scientist at the A-CIM Center and an adjunct assistant professor at the Center for Advanced Computer Studies (CACS) at the University of Louisiana at Lafayette, LA. His research interests are in the fields of virtual reality, intelligent robotic systems, Petri nets, and automated manufacturing systems. Dr. Gračanin is a member of IEEE, ACM, AAAI, APS, and SIAM.

Ignac Lovrek received his B.Sc., M.Sc., and Ph.D. degrees in Electrical Engineering from the University of Zagreb, Croatia, in 1970, 1973, and 1980, respectively. He is with the University of Zagreb where he is currently a Professor in the Department of Telecommunications, Faculty of Electrical Engineering and Computing, and the director of the Soft Telecommunication Technologies Laboratory. He led and participated in more than 10 national and international projects in the field of telecommunications, including the European Union Cooperation in Science and Technology project COST 247 Verification and Validation Methods for Formal Descriptions. He served as a program chair and a program committee member of several international conferences. He published two books and more than 100 technical papers. His research interests include call and service modeling and processing, telecommunication system architecture, software engineering and advanced telecommunications soft technologies. He is a member of the IEEE Communications Society, IEEE Computer Society, ACM, Euromicro and GI - Special Interest Group on Petri nets.

