A Framework for Multiuser Distributed Virtual Environments

Maja Matijasevic, *Member, IEEE*, Denis Gracanin, *Member, IEEE*, Kimon P. Valavanis, *Senior Member, IEEE*, and Ignac Lovrek, *Member, IEEE*

***Abstract*—A framework for multiuser distributed virtual environments (DVEs) has been proposed. The proposed framework, incorporating two models, the functional model and the interconnection model, attempts to represent common functionality, communication issues, and requirements found in multiuser DVEs. The functional model concentrates on the DVE functionality, while the interconnection model concentrates on how the components are interconnected to realize the required functionality. The models have been specified using the Unified Modeling Language. An experimental case study demonstrates the applicability and generality of the proposed approach.**

***Index Terms*—Distributed simulation, distributed virtual environment, modeling, networked virtual environment, virtual reality.**

I. Introduction

IN RECENT years, there has been a growing interest in the field of multiuser distributed virtual environments (DVEs). Superseeding the initial uses of DVEs for multiuser games and military and industrial team training, a new generation of applications has emerged, ranging from quite specific uses, for example, in medicine and robotics, to those of more general interest, such as interactive distance learning and on-line communities. Such DVEs are increasingly being developed based on open standards and delivered globally over the Internet [1], [2]. While there exist numerous technologies and application-specific strategies and techniques in design and implementation, the main motivation of this paper has been to provide a rather general framework for multiuser DVEs. The proposed framework is suitable for a variety of applications as it incorporates common functionality and communication issues found in DVEs.

Research and development in DVEs have mainly progressed in two complementary directions, one addressing the virtual reality end-systems' performance, the other addressing DVE communication. This paper is oriented toward the latter issue. It is postulated that DVE communication affects both the network traffic characteristics and functional aspects. While the first aspect has been well recognized and studied by traditional network traffic analysis and performance evaluation methods, the second aspect, functionality, has until recently received little attention, thus representing an open research issue. In this context, *functionality* means what the user "does" in the VE, and how this is interpreted within the application. This becomes a decisive issue when it comes to evaluating the quality of service as perceived by the user, i.e., in subjective and qualitative terms, rather than in objective and quantitative terms. Taking into account the above considerations, a study of network performance represents only a part of the overall performance evaluated by the user, while the interpretation of user-level interactions is transparent to the network. To address the above issues, the proposed framework considers two aspects of a VE system:

- the *application aspect* that relates to the question: How does an interaction at the application (user) level affect the communication characteristics?
- the *communication aspect* that relates to the question: How are events at the communication level reflected at the application level?

The application and the communication aspect of a DVE are considered as two "dimensions" of VE categorization, and treated as such, the proposed framework starts from the simplest case of a single-user local VE leading to the most general and complicated case of multiuser distributed VE.

The proposed general framework is based on two models, the functional and the interconnection models, that represent two complementary, interrelated views of a VE. Both models are presented using the Unified Modeling Language (UML) [3]. The functional model represents a DVE by identifying the necessary functionality for a specific application to meet its overall design purpose, while the interconnection model represents a distribution-oriented view of a DVE and highlights the different requirements for replication of shared objects versus streaming of continuous media. The elements of the functional model are grouped into user input and display, computing, and network input and output. The interconnection model, on the other hand, addresses the distribution of users, processes, and data in relation to group management, replication management, and group communication involving different types of media. A detailed description of functions in the framework, as well as the transition from the single-user local to the multiuser distributed VE, may be found in [4].

The framework presented in this paper may be applied for description and comparison of different implementations and

Manuscript received April 24, 1999; revised August 27, 2000 and December 28, 2001. This work was supported in part by the U.S. National Science Foundation under Grant BES-9506771 and Grant BES-9712565, and by the Board of Regents Support Fund Grant 98-00-ENH-TR-92R. This paper was recommended by Associate Editor R. Rada.

M. Matijasevic and I. Lovrek are with the Faculty of Electrical Engineering and Computing, University of Zagreb, HR-10000 Zagreb, Croatia (e-mail: maja.matijasevic@fer.hr; ignac.lovrek@fer.hr).

D. Gracanin is with the Computer Science Department, Virginia Tech, Falls Church, VA 22043 USA (e-mail: gracanin@vt.edu).

K. P. Valavanis is with the Department of Production Engineering and Management, Technical University of Crete, 73100 Chania, Crete, Greece (e-mail: kimonv@dpem.tuc.gr).

Publisher Item Identifier S 1083-4419(02)04378-9.

TABLE I
SUMMARY OF DVE NETWORKING ISSUES

	Distributed Virtual Environment						
Parameter	NPSNET	CAVE VPS	DIVE/COVEN	MASSIVE	mWorld	Spline	VLNET
Bandwidth							
- 3D world	yes	yes	yes	yes	yes	yes	yes
- text	no	no	yes	yes	no	no	yes
- audio	no	yes	yes	yes	yes	yes	yes
- video	no	yes	no	yes	no	no	yes
Latency	100/300 ms	100 ms	200 ms	n/a	n/a	n/a	100 ms
Distribution	multicast	multicast	multicast	multicast	multicast	multicast	unicast
Reliability	no	no	yes	no	yes	yes	yes
Network protocol	IPmc	IPmc	SRM/IPmc (nack)	IPmc	TCP/IP	SRM/IPmc	TCP/IP

may also assist in specification and development, as demonstrated in the case study. Thus, the main contribution of the paper is the framework itself, with specific accomplishments the two derived models (the functional and the interconnection model). While this paper emphasizes the theoretical foundations of the research, a complete implementation and experimental case study related to telerobotics has been published in [5].

The paper is organized as follows. Section II reviews related work. Section III gives an overview of the proposed framework, and Sections IV and V describe the functional model and the interconnection model, respectively. Section VI describes the implementation case study and summarizes the results. Section VII concludes the paper.

II. RELATED WORK

The development of the proposed framework has involved a study of research and work in the field. The review of related work was oriented toward multiuser DVEs, and especially toward interactive applications involving different multimedia components, such as three-dimensional (3-D) graphics, two-dimensional (2-D) graphics, audio, video, text, etc., as well as multiple senders and multiple receivers.

The first successful implementation of a large-scale real-time DVE was the SIMNET project, funded by U.S. Department of Defense in the early 1980s [6]. In the academic research community, multiuser DVEs began to be developed starting in the 1990s, using application-specific approaches over standard and experimental Internet protocols. In some ways, a successor of SIMNET, the Naval Postgraduate School Networked Vehicle Simulator (NPSNET)[7], already in its fifth release [8], was originally developed for battlefield simulation. The NPSNET group is also involved in the development of the virtual reality transfer protocol (vrtp) [9]. The Distributed Interactive Virtual Environment (DIVE) [10], developed at the Swedish Institute of Computer Science, started as a DVE for collaboration and teleconferencing. It has been further developed under European Advanced Communications Technologies and Services ACTS Project AC040 Collaborative Virtual Environments (COVEN) [11], the objective of which was to develop a general purpose platform for VE-based commercial applications for professional and home use. The Model, Architecture and System for Spatial Interaction in Virtual Environments (MASSIVE), developed at the University of Nottingham, U.K., is also a collaborative multiuser DVE system [12], [13]. The developers of MASSIVE are also known for developing AVIARY [14] and PARADE [15]. mWorld [16] is a multiuser collaborative DVE geared toward real-time 3-D editing and animation. The Scalable Platform for Large Interactive Networked Environments (SPLINE) [17], [18], by Mitsubishi Electric Research Lab, has been used to develop Diamond Park VE, with bicycling and social interaction as its main topic. MIRALab's Virtual Life Network (VLNET) [19] attempts to create realistic virtual humans. Several immersive DVEs use CAVE [20]; one such DVE is the Virtual Prototyping System [21] for collaborative vehicle design.

The multiuser DVEs listed above have been reviewed and compared based on a set of common parameters, as described in the sequel. It may also be noted that the above list of DVEs is by no means exhaustive or exclusive; there exist many other academic and research, as well as commercial, implementations. We consider selected DVEs not in order to compare different technologies, strategies, and techniques in DVE design and implementation, but to "extract" common functionality, requirements, and communication issues found in such DVEs.

For the purpose of this review, the requirements of DVEs have been divided in two broad categories: the networking support and the multiuser support. Networking support may be specified in terms of networking quality of service (QoS) parameters [22]. Bandwidth, latency, distribution scheme, and reliability have been selected for comparison of approaches applied in reviewed DVEs. In general, DVEs require high bandwidth to support multiple users and multimedia-rich VE content. The multiuser support leads to multipoint communication/distribution schemes [23]. As for VE content, it usually incorporates not only static and/or animated 3-D graphics primitives and models [24], but also other (multi)media types, such as images, audio, and video. Some data (pre-)processing techniques may be utilized to decrease the overall amount of transferred data, for example, behaviors and scripting, dead reckoning, filtering of updates based on spatial model of interaction, and data compression [25].

Also crucial for human performance in DVEs is the "real-time" requirement, i.e., the ability to satisfy human perception criteria for realism and fidelity [26]. Latency requirements, expressed in terms of total latency (delay, lag), and latency variance (jitter), critically affect human performance in DVEs [27]. The effects of user-observed latency can sometimes be compensated by buffering, predictive modeling, dead-reckoning algorithms, etc. [28]. The latency variance, in addition to degrading the realism of user's experience, can cause inconsistency in DVEs due to delayed or late updates of concurrent actions. The review of networking support issues has been summarized in Table I.

TABLE II
SUMMARY OF DVE MULTIUSER ISSUES

	Distributed Virtual Environment						
Parameter	NPSNET	CAVE VPS	DIVE/COVEN	MASSIVE	mWorld	Spline	VLNET
Purpose	military training	various, training	collaborative, prototyping	collaborative conferencing	collaborative	social, cycling	collaborative, games, shopping
Number of users	>1000	10 (per one CAVE)	20	60	n/a	n/a	8
Data model	replicated	shared, distributed	shared, distributed	replicated	shared, distributed	shared, distributed	shared, distributed
Data replication	full	full	partial	full	full	full	partial
Consistency	weak	weak	weak	weak	strong	weak	weak
Initialization	initialization database	initialization file	"closest peer"	state transfer from master agent	get copy of workspace	initial world	initial world
Join/leave	transparent	explicit request	transparent	explicit	explicit authorization	explicit	explicit
Access control	none	one input	none	soft, hard, and control locking	token passing	object ownership	server controlled
Awareness	spatial (AOIM, Area of Interest Manager)	none	spatial (aura, focus, nimbus)	spatial (locales, boundaries, aspects)	no	spatial (locales)	spatial (rooms, portals, AOIM)
Dead reckoning	yes	no	yes	no	no	"smooth motion"	yes

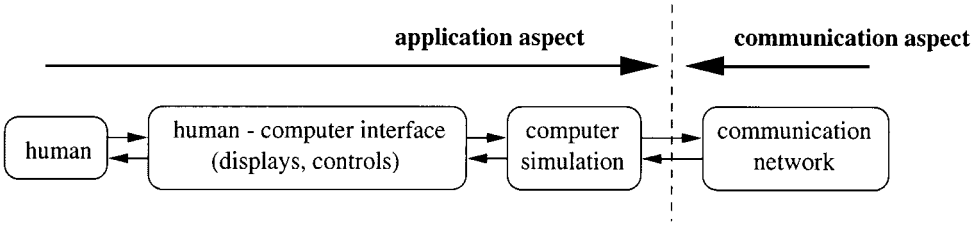

Fig. 1. Application and communication aspects of a VE system.

Regarding multiuser support [29], the main problem is maintaining consistency of a VE. Adequate mechanisms are needed for access control and concurrent interactions control [15]. Multiuser support issues for approaches applied in reviewed VEs are summarized in Table II.

The common issues for multiuser DVEs are incorporated in the framework presented in the next section.

III. FRAMEWORK OVERVIEW

The proposed framework incorporates the functional and the interconnection models of a multiuser DVE. The application and the communication aspects, shown in Fig. 1 with respect to DVE system components [30], have been used as motivation. The interdependence of the two aspects is a result of combining the virtual reality computing and networking technologies, which is the key characteristic of DVEs. Further on, the application and communication aspects may be considered as two dimensions of VE categorization, as illustrated in Fig. 2, with the number of users (single user versus multiuser) and the distribution of users and resources (local versus distributed) as the most salient features.

The simplest category is a single-user VE, implemented locally. Two orthogonal directions correspond to multiuser local VE and single-user DVE. The applications of interest follow the resultant direction, distributing both the users and resources

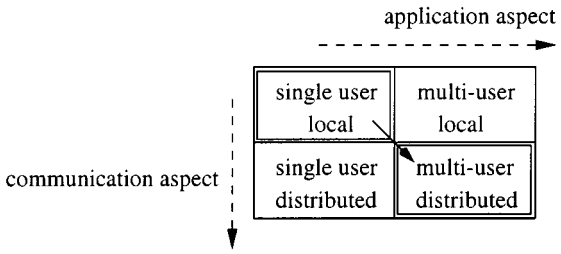

Fig. 2. "Dimensions" of a DVE.

over the network. In order to illustrate this transition, we start from identifying the functionality needed for a single-user local case, and expand it toward the multiuser distributed case, thus, building a general overall framework.

The functional and the interconnection models represent two complementary interrelated views of a multiuser DVE, as shown in Fig. 3. Such DVE may be viewed as a collection of (multiple) user processes and any number of autonomous (computer simulation) processes that communicate over the network, represented by the ***Network I/O*** block in the model. The functionality associated with each user's process involves human–computer interface and a computer simulation. The functional model hides the distributed nature of the overall VE behind the ***Network I/O***. Complementary to the functional model, the interconnection model addresses the distributed view, and expands

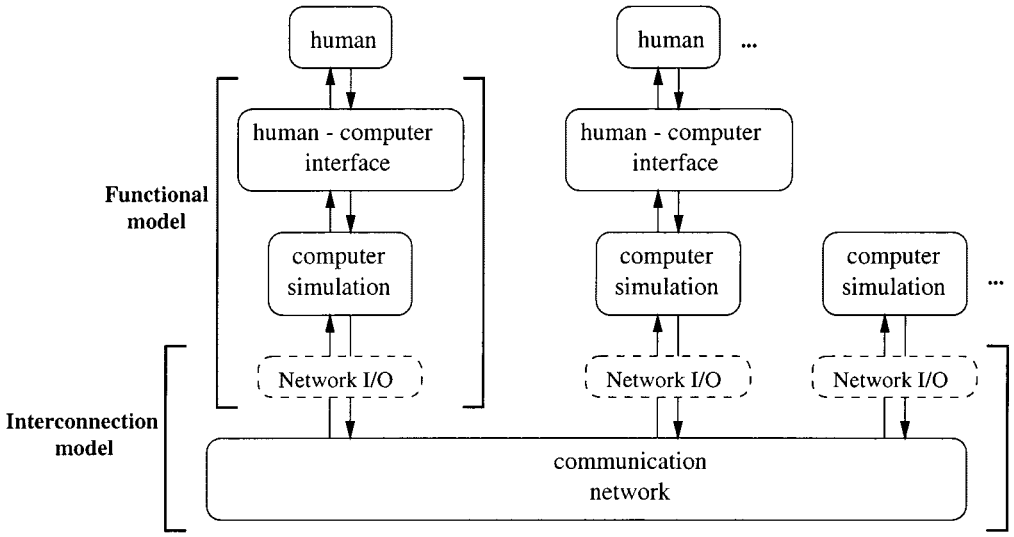

Fig. 3. Concept of two complementary models.

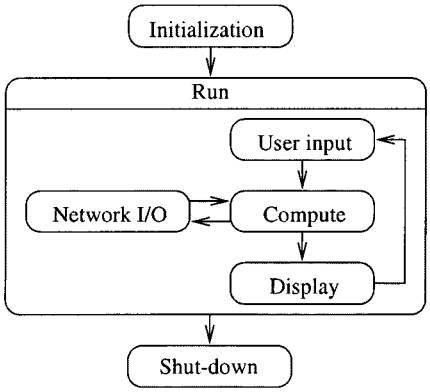

Fig. 4. General VE model.

on the ***Network I/O***, without distinction between user processes and autonomous processes.

The models are presented using the UML, which defines several types of diagrams for graphical notation. In our description of the functional model, *statechart diagrams* are used to describe the behavior of the system, and *activity diagrams* are used for activities representing the operations and sequencing of related activities. In the description of the interconnection model, the *class diagrams* are used for static structure description, and the *sequence diagrams* and *collaboration diagrams* are applied for description of interactions between UML components.

IV. Functional Model

The functional model represents a DVE application by identifying the necessary functionality for an application to meet its overall design purpose. An initial set of functions required to support a *single-user local VE* is defined first, and subsequently expanded and refined toward the most general *multiuser DVE* case.

A. Model Considerations

The initial functional model is depicted in Fig. 4. The main three states in the model are named ***Initialization***, ***Run***, and ***Shut-down***. ***Run*** corresponds to processing in each user's local VE application. It consists of four groups of functions that represent the core of the VE, namely, ***User input***, ***Compute***, ***Display***, and ***Network I/O***. The user is included in the loop between ***Display*** and ***User input***. ***Initialization*** involves initialization of all processes in ***User input***, ***Compute***, ***Display***, and ***Network I/O***, as well as instantiating the initial virtual world, before the control is passed to ***Run. Shut-down*** represents an orderly conclusion of the application, i.e., terminating all processes, as well as optional saving of state variables to be restored later.

B. Model Description

The functional model for the multiuser distributed VE is shown in Fig. 5. To accommodate different multiuser interactions, ***User input*** is divided in two components: ***User–VE***, which includes navigation, user–object, and user–user interactions, and ***User–group***, which includes membership and participation related interactions. ***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***. ***Virtual world management*** deals primarily with users' actions and the effects thereof within the VE, while ***Session management*** focuses on users' membership, participation, and communication between distributed processes that constitute the VE.

1) Virtual World Management: This 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 an 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 (mass 1 kg), and

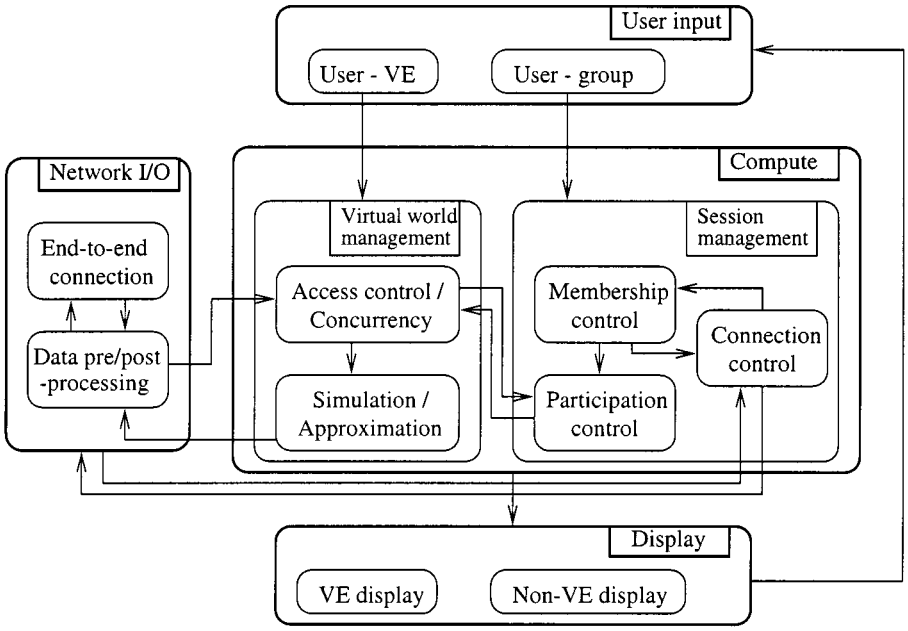

Fig. 5. Functional model.

spatial attributes that describe its position and orientation. Behaviors may be divided in two broad categories as deterministic and nondeterministic. Behaviors triggered by user actions are typically nondeterministic. A formal description of behaviors (not to mention standardization) is still an open research issue. Attributes and behaviors of an object are changed by the simulation, based on passage of simulation time and processing of the user input. The activity diagram for ***Simulation***, shown in Fig. 6, shows that ***Simulation***, in general, may be any combination of a ***Visual simulation***, an ***Audio simulation***, and a ***Physical simulation***.

The ***Simulation/Approximation*** expands the functionality of ***Simulation*** in the single-user local case. In a distributed VE, the control over the 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***. For example, ***Approximation*** may be implemented using a dead-reckoning algorithm. ***Approximation*** is tightly coupled with ***Simulation*** and is therefore represented in the same block. An appropriate approximation can typically be found that is 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's audiovisual properties need be known, and the scene is rendered based on updates received over the network and the approximation algorithm.

The amount of updates may be reduced by logical/functional partitioning, and declaring "area-of-interest managers" [7]. Another approach is to "filter" data based on virtual world physical properties, such as distance, or visual/aural occlusion [25], for limiting the scope of updates and transmissions only to interested parties. Both approaches, however, are applicable only in physically consistent VEs.

Access control/Concurrency implements the control of the local user's actions and concurrency restrictions due to multiuser access, serialization, and synchronization of multiple inputs, necessary for preserving consistency. ***Access control/Concurrency*** 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 multiuser applications [31]. When a role-based policy is used, access control operates in conjunction with ***Participation control***, which deals with user-to-role assignment.

Concurrency control is implemented on top of access control, with the goal of preserving integrity of shared virtual worlds. 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).

2) Session Management: Session management has two aspects in distributed multiuser applications that both need be addressed in the model: the 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). This aspect will be addressed as the *group control*. The communication aspect of session management deals with separating the control needed during the data transfer from the data transfer itself. It includes different groups of tasks related to session administration, session configuration, establishment/release, and control of synchronized exchange of information between peers.

In the model, group control comprises ***Membership control*** and ***Participation control***. These functions are necessary for

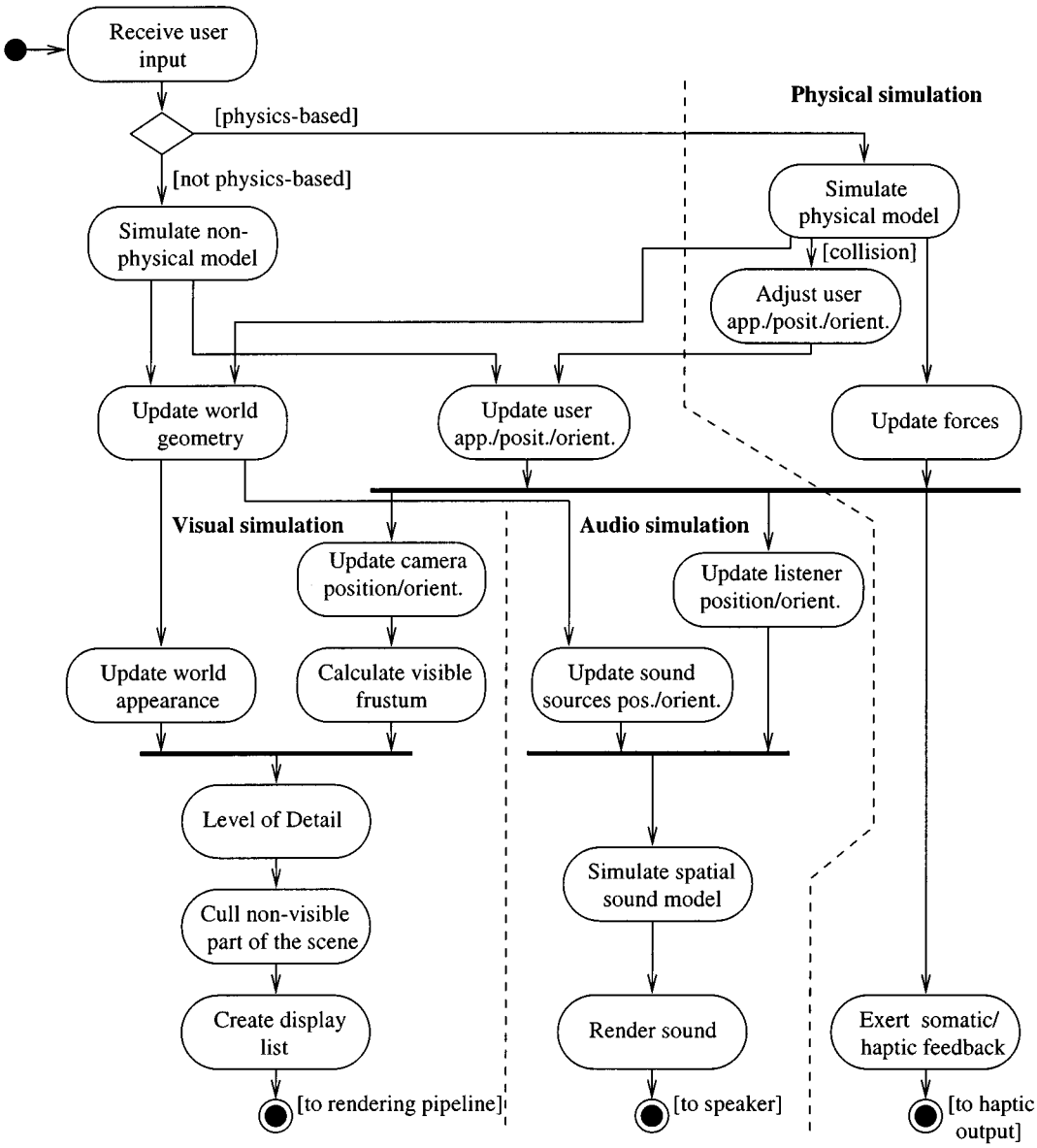

Fig. 6. Activity diagram for simulation.

accommodating both light-weight sessions and tightly coupled sessions, which differ in how the issues of membership, participation, and connection establishment are addressed. The ***Membership control*** enables a user to become a session member (join the shared application), or end its membership (leave the shared application). As mentioned earlier, the purpose of ***Participation control*** is to associate users with roles and to implement a participation policy. The user-to-role association may be either static (fixed for the duration of the session) or dynamic (renegotiable during the session). The link between ***Membership control*** and ***Connection control*** is through session establishment or release, as well as (re)negotiation of session QoS parameters.

3) Network I/O: This includes ***Data pre/post-processing***, and ***End-to-end connection***. ***Data pre/post-processing*** represents noncontext-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, is handled within ***Simulation/Approximation***, since only the 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 allows the users to choose what media types, and with what quality, they wish to receive. On the sending side, ***Data pre/post-processing*** acts as pre-processing, with the output written to ***End-to-end connection***. On the receiving side, ***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). ***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

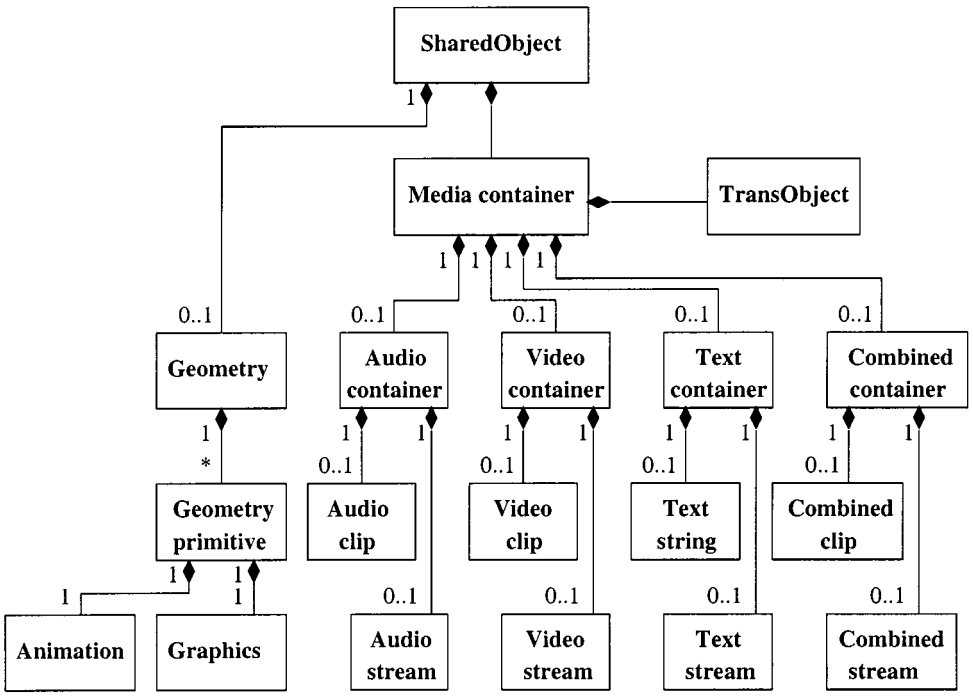

Fig. 7. SharedObject decomposition.

is performed by the session service, and 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.

V. Interconnection Model

The interconnection model represents a distribution-oriented view of a DVE. The analysis of DVE content and group communication highlight the different requirements for *replication of shared objects* versus *streaming of continuous media objects* at the transport level.

A classification of objects is proposed that divides objects into *SharedObjects* and *TransObjects*. A *SharedObject*, shown in Fig. 7, 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 a media container is to define spatial attributes of media element(s), for example, the position of a sound source in space. Two data types are distinguished for each medium, based on replication requirements: 1) stored media (clip), and 2) 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 the Web3D Consortium[1] Living Worlds Working Group and it is extended with terminology for continuous media objects.

A *TransObject* (Fig. 8) 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*).

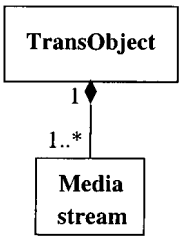

Fig. 8. TransObject decomposition.

Based on the above classification, the DVE transport requirements addressed in the interconnection model extend the fundamental requirements for continuous media by involving application aspects, such as 1) the manner in which media are composed within DVE content, and 2) user's perception of, and interaction with, DVE content.

A. Model Considerations

Distribution of users is an intrinsic property of a multiuser DVE. The session model for multiuser DVEs is based on the concept of *group* (or *multipoint*) *communication* [32], where a group corresponds to a set of participating processes, as shown in Fig. 3. One of the main concerns for a DVE is to maintain consistency, while allowing multiple user interactions. Three components may be identified as necessary for a group system support: group management, group communication, and replication management [33].

[1]Available: http://www.web3d.org

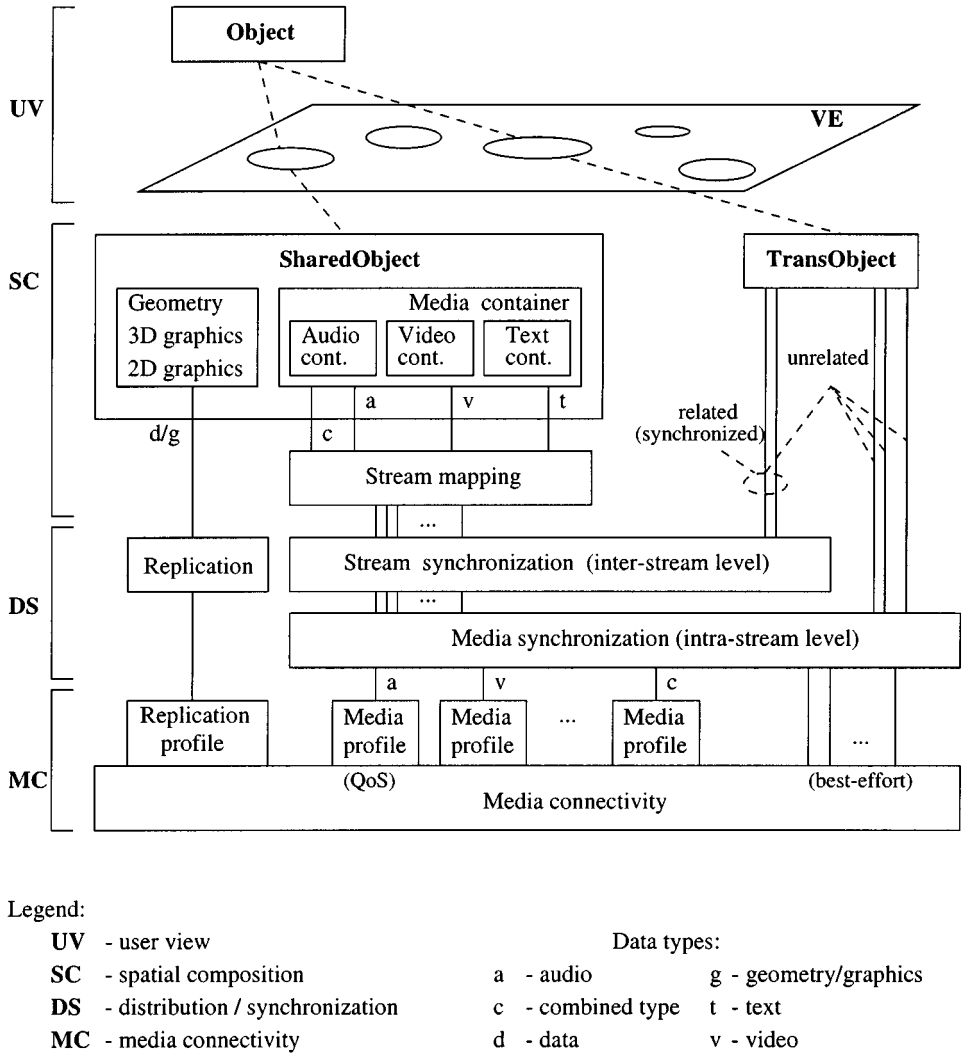

Fig. 9. Interconnection model.

Group management is needed to create and destroy a process group, and optionally to keep record of group membership that changes through join and leave requests. Group communication is a mechanism for information exchange among group members. The replication management uses group communication to maintain consistency between data copies associated with multiple processes within a group.

B. Model Description

Fig. 9 illustrates the interconnection model. The model shows mapping of virtual world objects from the user view (UV) to media connectivity (MC). A UV corresponds to an individual user's experience of the shared virtual world. In the user view, a DVE is a collection of virtual world objects, including the environment, static and dynamic objects, and embodiments of other users or autonomous processes (agents). In this view, the actual 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 an 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 exists a temporal relationship between replicas on distributed hosts. In the model, this is shown as replication associated with *SharedObjects*. A *TransObject* contains references to multiple media streams, that may be either unrelated or temporally related in a common presentation. Intra- and interstream synchronization is needed for all such streams, regardless of whether a stream is mapped to a media container in a *SharedObject*, or presented directly to the user as *TransObject*. Interstream synchronization typically requires less processing if the stream encoding already contains synchronization in-

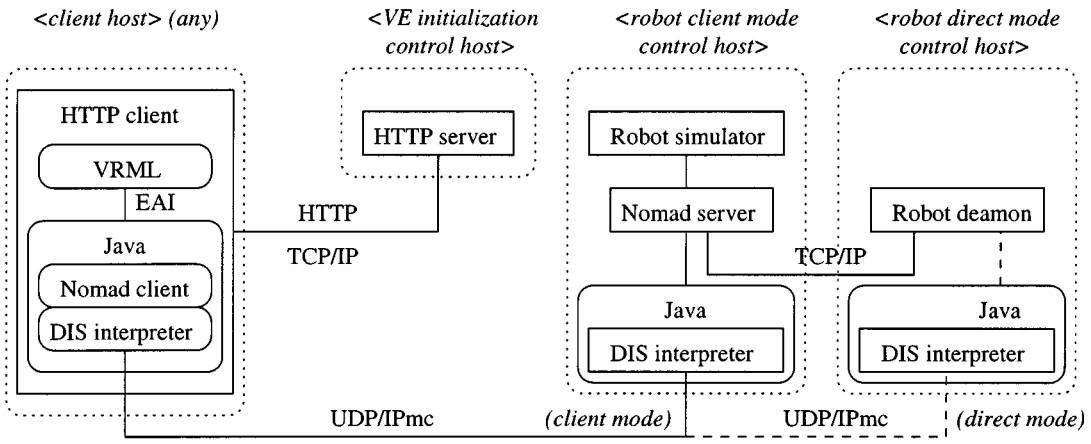

Fig. 10. Implementation of a multiuser distributed VE.

formation (combined type). Otherwise, 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 cannot be determined by the model.

The media connectivity (MC) level deals with mapping of application level QoS parameters to transport parameters. 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 also depend on stream encoding. For example, both uncompressed audio and video tolerate some loss, but they do 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 a per-connection basis.

VI. Case Study and Results

The case study used to demonstrate the application of the proposed framework is a multiuser DVE for teleoperation of a mobile robot. The experimental data from the implementation is intended to illustrate the interdependence between network traffic and interactions at the user level. This DVE has been developed in the Robotics and Automation Laboratory (RAL) and the Virtual Reality and Multimedia Laboratory of the University of Louisiana at Lafayette. For further details of the implementation, the reader is referred to [5].

A. DVE Design and Development

The DVE presented in this case study has been implemented using a combination of standard components: Virtual Reality Modeling Language (VRML) [34], IEEE 1278 Distributed Interactive Simulation (DIS) [35], and Java programming language. The DIS–Java–VRML [36] approach is based on the following.

- VRML displays the 3-D virtual environment.
- Java provides support for networking and portability, and controls the VRML display.
- DIS is used for communicating DVE state changes (e.g., object position, orientation, velocity, etc.).

By complying with these standards, rather than being limited to specific hardware, a virtual world could potentially be viewed on any computer with a network connection and a Web browser.

It may also be noted here that DIS standard has been superseeded by the High Level Architecture (HLA), a more recent standard for simulation interoperability. The HLA has been specified in the IEEE Standard 1516 series. For the purpose of research described in this case study, however, DIS–Java–VRML has been an adequate solution. The implementation of DIS protocol in Java programming language is a work-in-progress of the DIS–Java–VRML Working Group of the Web3D Consortium.

The DVE has been developed based on the following requirements.

- Create a virtual model of the mobile robot and the laboratory environment.
- Create a virtual control panel to control the robot. The user who controls the robot (temporarily) owns this device.
- Integrate the robot control in the virtual environment and enable real-time user control (teleoperation) of the robot.
- Ensure exclusive control over the robot (i.e., only one user at a time may control the robot), while enabling multiple users to view the ongoing experiment.
- Enable transfer of control among different users as an extension of basic functionality.

The final implementation design is shown in Fig. 10. In addition to client host(s), there are three control components: the VE initialization control, the robot client-mode control, and the robot direct-mode control. The VE initialization uses a WWW browser for download of the initial virtual world (VRML file) and the virtual control panel (Java applet) that substitutes the original robot GUI (Fig. 11). The control panel contains buttons

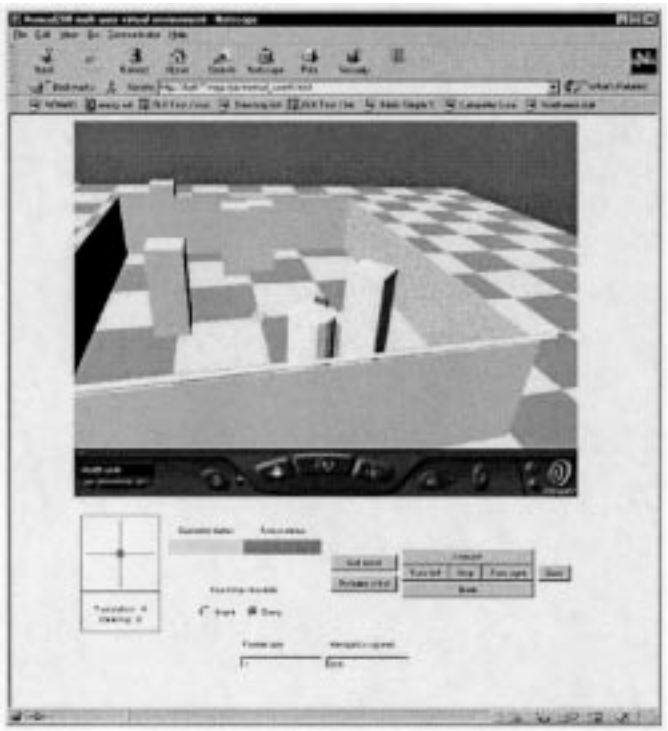

Fig. 11. Telerobotics DVE snapshot.

and "soft joystick" for robot teleoperation. The communication between the Java applet and the VRML browser uses the external authoring interface (EAI). The robot client-mode control uses a Nomad 200 mobile robot simulator, included in 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. To establish communication with the Nomad server from the virtual world on the client host, the Java applet implements the Nomad client, using the locally developed implementation of Nomad API in Java [37]. The multiuser environment uses DIS over IP multicast and also implements exclusive robot control in a multiuser environment. 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 protocol data units (PDUs) before multicasting them over the network. On the server side, 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. On the client side, the preferred solution has been to have the DIS interpreter run directly from the applet and write PDUs to the multicast port. That required overriding the security restriction imposed by Java API implementation which does not normally allow applets to use multicast sockets. The DIS interpreter at the Nomad server side runs as a standalone Java application that is not subject to applet multicast restriction. The functional and the interconnection model are used for the specification of the DVE; the mapping from the models to the case study is presented in detail in [4].

TABLE III
SUMMARY OF IMPLEMENTATION PERFORMANCE

PDU	Platform	Speed [PDU/s] instantiation	cloning	sending
Action Request (40 byte)	SGI	3003.00	3442.73	2866.42
	PC	3036.43	3906.25	8287.29
Action Request (56 byte)	SGI	1914.24	3503.03	2893.51
	PC	2202.64	3896.10	8823.52
Action Response (40 byte)	SGI	3026.63	3452.24	3003.00
	PC	3006.01	3836.31	8021.39
Entity State (144 byte)	SGI	1500.90	1073.34	3354.95
	PC	1628.66	1492.53	4398.82

B. DIS–Java–VRML Performance

The DIS standard is based on the concept of an interactive virtual environment consisting of autonomous simulation applications running on multiple networked computers. Moreover, DIS is intended to support a mixture of virtual entities (human-in-the-loop simulators), live entities (operational platforms and test and evaluation systems), and constructive entities (war games and other automated simulations). The standard consists of two parts. The application protocols part defines the format of standard DIS PDUs for communicating events between simulators. PDUs are grouped in several families, including entity information/interaction, logistics, simulation management, warfare, radio communications, and distributed emission regeneration. In practice, a simulation may use any subset of DIS PDUs. The communications services and profiles part of the standard follows the ISO Open System Interconnection (OSI) reference model, and specifies three classes of services (best-effort multicast, best-effort unicast, and reliable unicast) and performance parameters in terms of transport-to-transport latency, transport-to-transport acceptable reliability, transport-to-transport jitter, and transport-to-physical acceptable latency.

In the case study implementation, the following subset of DIS PDUs has been used. The entity state PDU (ESPDU) from the family of entity information/interaction PDUs has been used for replication, i.e., communicating the updated state of the virtual world objects, including position, orientation, and appearance. The access request and access response PDUs from the family of simulation management PDUs have been used for negotiation and transfer of access control to robot, as well as for teleoperation of the robot.

The DIS implementation performance, addressed first, evaluates the Java implementation of DIS in terms of PDU generation rate. The results show the rate of PDU generation, which consists of two elements: *PDU instantiation* or *PDU cloning*, and *PDU sending*.

Several measurements have been performed. The instantiation/cloning tests measure the time to instantiate a given number of PDUs of particular type. The send test instantiates a single PDU, serializes it into a datagram, and then repeatedly sends it for a given number of iterations. The tests have been performed using a SGI O_2 workstation and a Pentium PC. Average speeds for instantiation, cloning, and sending have been calculated on the basis of 5000 PDUs and summarized in Table III. Instanti-

ation and cloning tests demonstrate that the time for these two operations depends mainly on DIS PDU length.

For PDU sending, the average sending speed does not differ much for different PDU types, since all PDUs fit into one User Datagram Protocol (UDP) datagram. The overall time to generate a PDU is the sum of times for cloning/instantiation and sending time. For the largest PDU, the ESPDU, which may be considered as the worst case, the limits of the DIS–Java implementation using the hardware and software configuration in the testbed are in the range of 1500 PDU/s for instantiation/cloning and 3000–4000 PDU/s for sending.

C. Testbed Description

The actual testbed for measurements includes a Nomad 200 mobile robot base from Nomadic Technologies Inc., equipped with a shared memory multiprocessor control system. The low-level processors implement motion control and sensor processing modules, including tactile, ultrasonic, and basic vision system. The master processor is implemented using a Pentium-based PC (Pentium 133 MHz, 32 MB RAM), running Red Hat Linux release 3.0.3 (Picasso), Kernel 2.0.24. In the physical environment, the robot is set in the dedicated area in RAL, called the *Blue Room*. The robot uses a wireless Ethernet link to RangeLAN2/Access Point (from Proxim, Inc.) to communicate with other computers in RAL's 10 Mb/s Ethernet LAN. A Silicon Graphics O_2 workstation running IRIX 6.3 and Gateway 2000 personal computers (Pentium II 333 MHz, 384 MB RAM), running Microsoft Windows 95/NT/98, have been used as client hosts.

D. Multiuser Scenarios

The scenarios presented next demonstrate the use of the application with ten and 16 simultaneous users, and illustrate dependency of network requirements on the number and the behavior of users. Throughput, delay, and PDU statistics were measured. Due to multicast and software compatibility limitations in previous versions used in the testbed, scenarios M1–M4 include ten simultaneous users, and scenarios M5–M7 include 16 simultaneous users.

The components from Fig. 10 were set as follows: VE initialization control host, SGI O_2; client hosts, ten PCs and six workstations; robot client mode control host, SGI O_2; and robot direct mode control host, Nomad 200.

The `snoop` packet capture program, included as a part of the IRIX operating system distribution, was used for data collection. The collected data was analyzed using different filters to extract the packets of interest and examine the relevant packet fields (time stamps, DIS ESPDU header fields, etc.).

1) Scenario M1: Bandwidth Requirements for Inactive Users: For the purpose of this scenario, after initialization of the robot, users join the DVE at an (approximate) rate of 1 user/min. 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. Fig. 12 shows the multicast throughput measured in PDUs/s for controlled join of ten users. The throughput is directly proportional to the number of users (PDU sources),

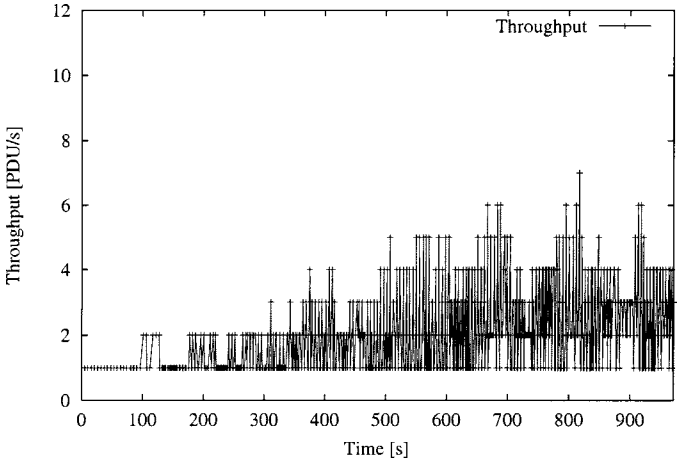

Fig. 12. Multicast throughput in scenario M1.

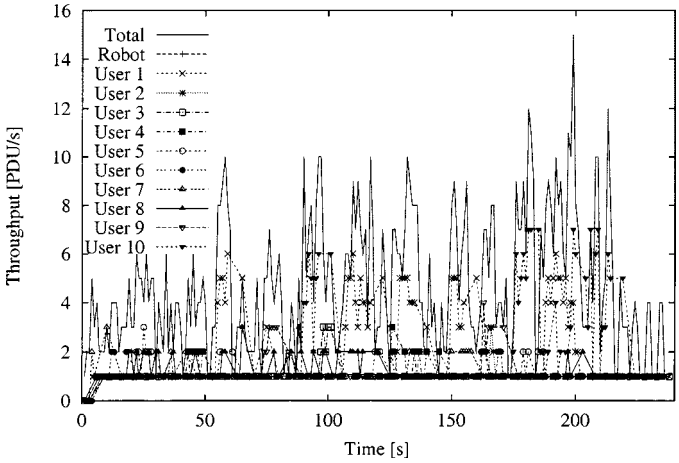

Fig. 13. Multicast throughput in scenario M2.

including the robot. All PDUs recorded in this scenario are ESPDUs, representing keep-alive messages, periodically generated by users and the robot.

2) Scenario M2: Bandwidth Requirements for Various Navigation Styles: Scenario M2 starts with ten users present in the DVE. Independently of each other, the users navigate through the DVE using the browser built-in commands (browser dashboard) and predefined viewpoints. Fig. 13 shows the overall multicast throughput, with the idea to demonstrate that, contrary to scenario M1, the contribution of individual user-generated traffic here depends on user's activity. Navigation is characterized by "bursts" of ESPDUs denoting the change of user's viewpoint position and/or orientation. All PDUs recorded in this scenario are ESPDUs since no exchange of control occurs.

3) Scenario M3: Unsuccessful Exchange of Control: In scenario M3, ten users are initially present in the DVE. One user acquires the control over the robot, thus becoming an operator. The default response to incoming 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. Fig. 14 shows the multicast throughput and delay between action request and action response PDUs. It may

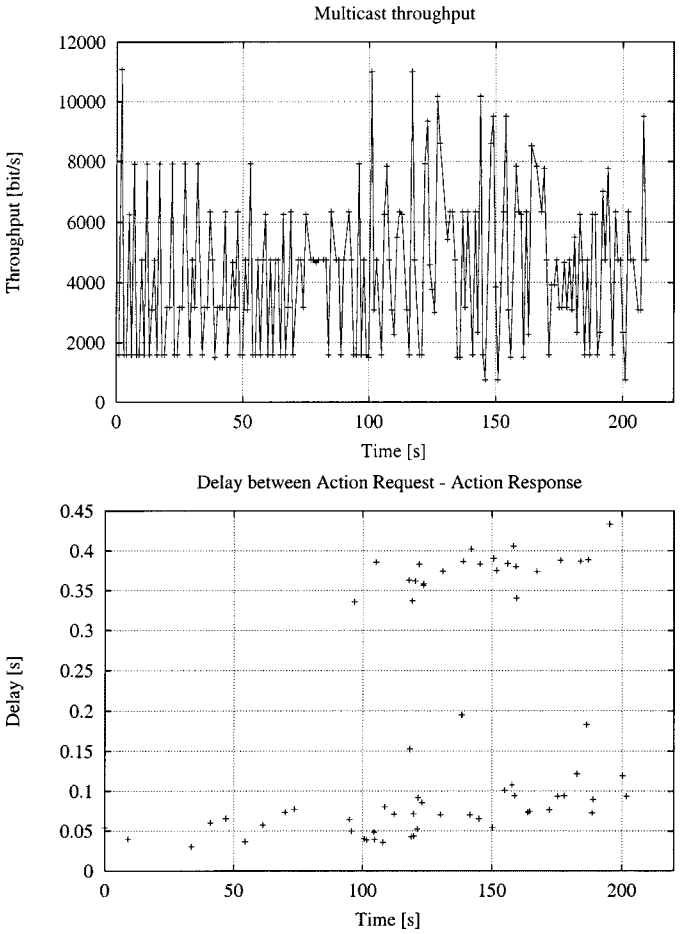

Fig. 14. Multicast throughput and delay in scenario M3.

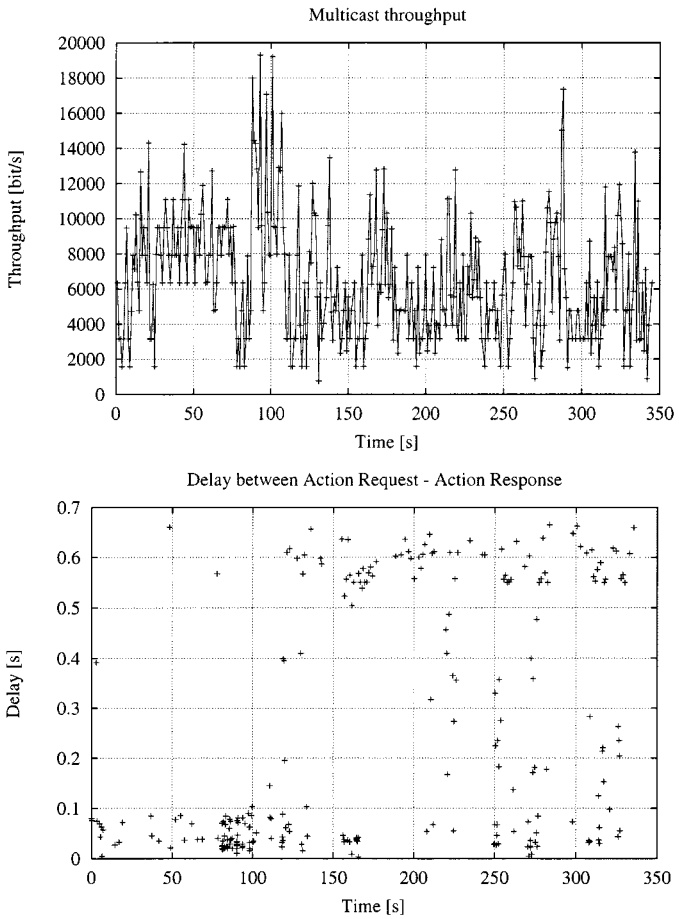

Fig. 15. Multicast throughput and delay in scenario M4.

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 ESPDUs (77.22%), and the rest are action request/action response (11.39%, 11.39%).

4) Scenario M4: Successful Exchange of Control: In scenario M4, ten users are initially present in the DVE. 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. Fig. 15 shows the multicast throughput and delay between action request and action response PDUs. PDU statistics for scenarios M3 and M4 are given in Table IV.

PDU statistics shows ESPDUs (68.8%) and a relative increase in action request and 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 four PDUs: the action request/action response pair, and two 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 average delay, due to additional PDU processing as well as potentially misdirected requests. The latter may happen when the exchange of control

TABLE IV
PDU STATISTICS FOR SCENARIOS 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%

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) Scenario M5: Multiple Activities and Button Teleoperation: Scenarios M5 and M6 study the "realistic" throughput with 16 simultaneous users. These scenarios are realistic in the sense that they are less restrictive regarding user's behavior; 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. The difference between scenarios M5 and M6 is in teleoperation activity. In scenario M5, teleoperation is performed using buttons, and in scenario M6, it is performed using the soft joystick.

In scenario M5, 16 users are initially present in the DVE. User navigation, as well as transfer of control, happens randomly, but the robot operator only uses buttons for robot teleoperation. This scenario may be considered as the lower bound of

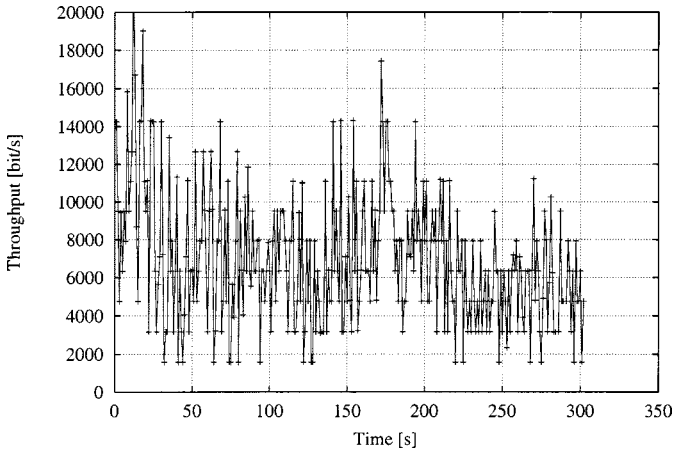

Fig. 16. Multicast throughput in scenario M5.

TABLE V
PDU STATISTICS FOR SCENARIOS 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%

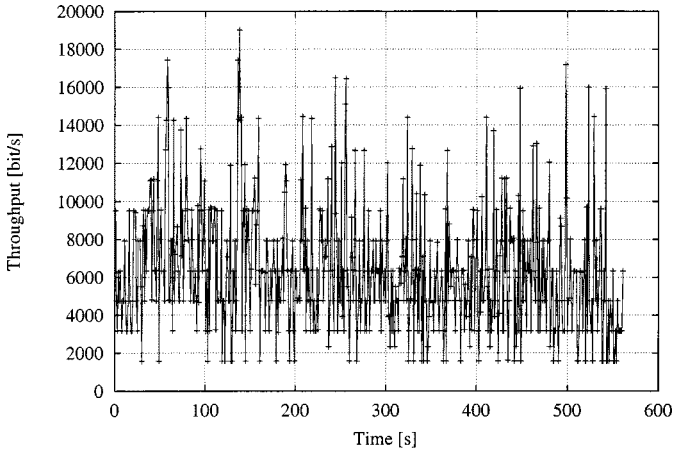

Fig. 17. Multicast throughput in scenario M6.

throughput during active participation. Fig. 16 shows the multicast throughput in scenario M5. Maximum throughput in this scenario is 22 176 b/s, and average throughput is 6974.12 b/s. PDU statistics in scenarios M5 and M6 are given in Table V. It may be noted that the total number of PDUs generated in scenario M6 is significantly higher than that in scenario M5.

6) Scenario M6: Multiple Activities and Joystick Teleoperation: In scenario M6, 16 users are initially present in the DVE. User navigation, as well as transfer of control, happens randomly, but the robot operator only uses the soft joystick for robot teleoperation. Fig. 17 shows the multicast throughput for scenario M6. Maximum throughput in this scenario is 19 008 b/s, and average throughput is 6362.13 b/s.

The results demonstrate the patterns in PDU statistics and multicast throughput for different types of navigation in the DVE, as well as in using different ways of controlling the robot.

VII. CONCLUSION

The presented framework, specified using UML, attempts to represent common functionality, communication issues, and requirements found in multiuser DVEs. The functional model concentrates on the DVE functionality, while the interconnection model concentrates on how the components are interconnected to realize the required functionality. A proof of concept has been verified using a telerobotics application development case study. Future work on the framework involves enhancements related to modeling of the autonomous process functionality, as well as DVE performance. Specifically, we are interested in including the sensor(s) feedback, which will "close the loop" between the simulator and the process running the control algorithm for the purpose of achieving robot autonomy.

REFERENCES

[1] J. Vince and R. Earnshaw, Eds., *Virtual Worlds on the Internet*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1998.
[2] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*. New York: ACM Press Siggraph Series and Addison-Wesley, 1999.
[3] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley, 1999.
[4] M. Matijasevic, "Distributed networked multimedia and virtual reality applications for multiuser environment," Ph.D. dissertation, Univ. Zagreb, Zagreb, Croatia, Dec. 1998.
[5] M. Matijasevic, K. P. Valavanis, D. Gracanin, and I. Lovrek, "Application of a multiuser distributed virtual environment framework to mobile robot teleoperation over the Internet," *Mach. Intell. Robot. Contr.*, vol. 1, no. 1, pp. 11–26, 1999.
[6] D. C. Miller and J. A. Thorpe, "SIMNET: The advent of simulator networking," *Proc. IEEE*, vol. 83, pp. 1114–1123, Aug. 1995.
[7] 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: Teleop. Virtual Environ.*, vol. 3, no. 4, pp. 265–287, 1994.
[8] M. Capps, D. McGregor, D. Brutzman, and M. Zyda, "NPSNET-V: A new beginning for dynamically extensible virtual environments," *IEEE Comput. Graph. Appl.*, vol. 20, no. 5, pp. 12–15, 2000.
[9] D. Brutzman, M. Zyda, K. Watsen, and M. Macedonia, "Virtual reality transfer protocol (vrtp) design rationale," in *Proc. WET ICE: Sharing a Distributed Virtual Reality*, Cambridge, MA, June 1997, pp. 179–186.
[10] O. Hagsand, "Interactive multiuser VEs in the DIVE system," *IEEE Multimedia*, vol. 3, no. 1, pp. 30–39, 1996.
[11] ——, (1999, Dec.) COVEN—Collaborative Virtual Environments. Final Report. ACTS Project Number AC040. [Online]. Available: http://www.infowin.org/ACTS/
[12] C. M. Greenhalgh and S. D. Benford, "Supporting rich and dynamic communication in large-scale collaborative virtual environments," *Presence: Teleop. Virtual Environ.*, vol. 8, no. 1, pp. 14–35, 1999.
[13] J. Purbrick and C. Greenhalgh, "Extending locales: Awareness management in MASSIVE-3," *Proc. IEEE Virtual Reality 2000 Conf.*, p. 287, 2000.
[14] D. N. Snowdown and A. J. West, "AVIARY: Design issues for future large-scale virtual environments," *Presence: Teleop. Virtual Environ.*, vol. 3, no. 4, pp. 288–308, 1994.
[15] D. J. Roberts and P. M. Sharkey, "Minimizing the latency induced by consistency control within a large-scale multiuser distributed environment," *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 5, pp. 4492–4497, 1997.
[16] J. M. S. Dias, R. Galli, A. C. Almeida, C. A. C. Bello, and J. M. Rebordão, "mWorld: A multiuser 3-D virtual environment," *IEEE Comput. Graph. Appl.*, vol. 17, no. 2, pp. 55–65, Mar.–Apr. 1997.
[17] D. B. Anderson, J. W. Barrus, J. H. Howard, C. Rich, C. Shen, and R. C. Waters, "Building multiuser interactive multimedia environments at MERL," *IEEE Multimedia*, vol. 2, no. 4, pp. 77–82, 1995.
[18] J. W. Barrus, R. C. Waters, and D. B. Anderson, "Locales: Supporting large multiuser virtual environments," *IEEE Comp. Graph. Appl.*, vol. 16, no. 6, pp. 50–57, Nov. 1996.

[19] T. K. Çapin, I. S. Pandzic, N. Magnenat-Thalmann, and D. Thalmann, *Avatars in Networked Virtual Environments*. West Sussex, U.K.: Wiley, 1999.
[20] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The CAVE: Audio visual experience automatic virtual environment," *Commun. ACM*, vol. 35, no. 6, pp. 65–72, June 1992.
[21] V. D. Lehner and T. A. DeFanti, "Distributed virtual reality: Supporting remote collaboration in vehicle design," *IEEE Comput. Graph. Appl.*, vol. 17, no. 2, pp. 13–17, Mar.-Apr. 1997.
[22] "Information Technology—Quality of Service—Framework ISO/IEC DIS 13 236," International Standards Organization, Geneva, Switzerland, 1997.
[23] P. Bagnall, R. Briscoe, and A. Poppitt, "Taxonomy of communication requirements for large-scale multicast applications,", Internet RFC 2729, Dec. 1999.
[24] M. R. Macedonia and M. J. Zyda, "A taxonomy for networked virtual environments," *IEEE Multimedia*, vol. 4, no. 1, pp. 48–56, Jan.–Mar. 1997.
[25] B. Roehle, "Channeling the data flood," *IEEE Spectrum*, vol. 34, no. 3, pp. 32–38, Mar. 1997.
[26] M. R. Stytz, "Distributed virtual environments," *IEEE Comput. Graph. Appl.*, vol. 16, no. 3, pp. 19–31, May 1996.
[27] K. S. Park and R. V. Kenyon, "Effects of network characteristics on human performance in a collaborative virtual environment," *Proc. IEEE Virtual Reality Conf.*, pp. 104–111, Mar. 1999.
[28] M. Zyda, Ed., *Modeling and Simulation: Linking Entertainment and Defense*. Washington, DC: National Academy Press, 1997.
[29] C. A. Ellis, S. J. Gibbs, and G. L. Rein, "Groupware: Some issues and experiences," *Commun. ACM*, vol. 34, no. 1, pp. 38–58, Jan. 1991.
[30] N. I. Durlach and A. S. Mavor, Eds., *Virtual Reality: Scientific and Technological Challenges*. Washington, DC: National Academy Press, 1995.
[31] W. K. Edwards, "Policies and roles in collaborative applications," in *Proc. ACM Conf. Computer Supported Cooperative Work*, Boston, MA, 1996, pp. 11–20.
[32] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint communication: A survey of protocols, functions, and mechanisms," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 277–290, Apr. 1997.
[33] F. J. N. Cosquer and P. Veríssimo, "Survey of selected groupware applications and platforms," ESPRIT, Basic Research Project 6360: BROADCAST Tech. Rep. 40, 1994.
[34] ——, "Information Technology—Computer graphics and image processing—The Virtual Reality Modeling Language (VRML)—Part 1: Functional specification and UTF-8 encoding. ISO/IEC 14 772-1," International Standards Organization, Geneva, Switzerland, 1997.
[35] ——, IEEE Standard for Distributed Interactive Simulation, Std. 1278-1995, 1278-1998, IEEE, Inc., New York, Apr. 5, 1996.
[36] D. Brutzman, "The Virtual Reality Modeling Language and Java," *Commun. ACM*, vol. 41, no. 6, pp. 57–64, 1998.
[37] D. Gracanin, M. Matijasevic, A. Chandrupatla, P. Subramaniam, and S. L. V. Ippagunta, "Integration of mobile robot into virtual reality testbed," in *Proc. 1998 IEEE Int. Conf. Control Applications*, vol. 2, Trieste, Italy, 1998, pp. 1302–1306.

Maja Matijasevic (M'95) received the Dipl.-Ing. degree in 1990, the M.Sc. degree in 1994, and the Ph.D. degree in 1998, all in electrical engineering, from the University of Zagreb, Croatia, and the M.Sc. degree in computer engineering in 1997 from the University of Louisiana at Lafayette.

She is currently an Assistant Professor in the Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb, which she joined in 1991. She was a Research Associate in the Virtual Reality and Multimedia Laboratory of the A-CIM Center at the University of Louisiana at Lafayette, from August 1996 to April 1999, and a Visiting Researcher at the Technical University of Graz, Austria, on several occasions between 1992 and 1996. Her main research interests include distributed virtual environments and advanced multimedia services for new generation networks.

Dr. Matijasevic is a member of the Association for Computing Machinery and the Upsilon Pi Epsilon Honor Society in the Computing Sciences.

Denis Gracanin (M'88) was born in Rijeka, Croatia, on May 12, 1963. He received the B.S. and M.S. degrees in electrical engineering from the University of Zagreb, Croatia, in 1985 and 1988, respectively, and the M.S. and Ph.D. degrees in computer science from the University of Louisiana at Lafayette in 1992 and 1994, respectively.

He was a Research Scientist in the A-CIM Center, University of Louisiana at Lafayette, from January 1994 to August 1999, and an adjunct Assistant Professor in the Center for Advanced Computer Studies, University of Louisiana at Lafayette, from August 1997 to August 1999. Since August 1999, he has been an Assistant Professor in the Computer Science Department, Virginia Tech, Falls Church. His current research interests include distributed virtual environments, distributed simulations, and Internet computing.

Dr. Gracanin's professional memberships include the Association for Computing Machinery, AAAI, SCS, and SIAM. He is also a Professional Engineer in electrical engineering, Louisiana Professional Engineering and Land Surveying Board. In 1991, he received a Fulbright scholarship, from August 1991 to May 1992, for studies at the University of Louisiana at Lafayette.

Kimon P. Valavanis (S'80–M'83–SM'91) 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.

He is currently with the Technical University of Crete serving as Professor in the Department of Production Engineering and Management, Director of Graduate Studies, and Director of the Robotics and Intelligent Systems and Technology Laboratories. He is also a Guest Professor in the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, and an Adjunct Professor at The Center for Advanced Computer Studies at the University of Louisiana at Lafayette. His research interests are in the areas of robotics, automation, and distributed intelligence systems. He has authored or coauthored over 200 refereed publications. He has organized and taught Tutorial Workshops at the IEEE CDC, ACC, IEEE ICRA, and Conference on Telecommunications ConTEL. He serves on the Editorial Advisory Board of the International Series on Microprocessor Based and Intelligent Systems Engineering Series, published by Kluwer Academic Publishers.

Dr. Valavanis has been on the organizing committee of many IEEE conferences, serving as General, Program, Registration, and Local Arrangements Chair. Since January 1996, he has been the Editor-in-Chief of the *IEEE Robotics and Automation Society Magazine*. In 1998, he was elected as Vice President—Administration of the IEEE Mediterranean Control Association. He is a former Fulbright Scholar, and a Distinguished Speaker in the IEEE Robotics and Automation Society.

Ignac Lovrek (M'95) received the 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 currently a Professor in the Department of Telecommunications of the Faculty of Electrical Engineering and Computing at the University of Zagreb, Croatia. His research interests include call and service modeling and processing, telecommunication system architecture, software engineering and new soft technologies for telecommunications. He leads a research group working on the project Call and Service Processing for Advanced Network Architectures funded by the Croatian Ministry of Science and Technology, and related industrial projects. He serves as a member of editorial board of the *International Journal of Knowledge-Based Intelligent Engineering Systems* and the series Advances in Computing published by Springer, Wien, Austria, and New York.

Dr. Lovrek is a member of the Association for Computing Machinery, EUROMICRO and Special Interest Group on Petri Nets and Related System Models, Gesellschaft für Informatik, Germany.