MVIP
- Audio Enabled Multicast VNet

John L. Robinson, John A. Stewart and Isabelle Labbe
Communications Research Centre Canada *

ABSTRACT

This paper presents a multicast approach to shared virtual worlds. A shared VRML world is described with integrated spatial audio in a freeware VRML browser. An implementation in Linux of multicast FreeWRL with the Robust Audio Tool (RAT) is presented. To support this audio enabled multicast VRML prototype, MVIP (Multicast VRML Interchange Protocol) is implemented as a Java program using the services of the Real-Time Protocol (RTP) and accessing VRML via the External Authoring Interface (EAI).

CR Categories and Subject Descriptors: C.2.2 [Computer Communication Networks]: Network Protocols; C.2.4 [Computer Communication Networks]: Distributed Systems – Distributed Applications; H.5.1 [Information Interfaces and Presentation] Multimedia Information systems – Artificial, Augmented and Virtual Realities; H.5.3 [Information Interfaces and Presentation] Collaborative Computing.
Additional Keywords: Virtual reality markup language (VRML), distributed virtual environments, multi-user virtual reality, IP multicasting, MBone

1 INTRODUCTION

Interest in shared virtual worlds has been evident for some years, notably in the development of Distributed Interactive Simulation (DIS) and the High Level Architecture/Run Time Infrastructure (HLA/RTI) by the Defense Advanced Research Projects Agency (DARPA) and the US Department of Defense. At the same time VRML has emerged as an approach for the 3-D Web and in parallel methods to share such a virtual world are being developed [17]. To contribute to the development of shared VRML, building on concepts from the client-server based VRML interchange protocol used by the VNet [15], a shared multicast virtual world protocol is proposed. The approach is seen as complementary to the dis-Java-vrml and virtual reality transfer protocol (VRTP) work [16], with the emphasis here on service to a relatively small community of participants and on ease in supporting the amalgamation of multicast applications, for example a shared VRML world with talking avatars.

This paper introduces a shared virtual world based on our multicast VRML Interchange Protocol (MVIP) and describes a method for encapsulating the MVIP information in RTP. In an earlier project [6] we had developed a shared VRML viewer using FreeWRL [8] and the client-server VNet on a Linux platform. Our experimental implementation is built on that foundation, using shared FreeWRL in combination with multicast audio (RAT).

In the next section there is a brief discussion to place this work in the context of our research interests and of other research activities in shared VRML worlds. After that we introduce Multicast VRML Interchange Protocol (MVIP) implemented as a Java program that uses the services of RTP and accesses VRML via the EAI. The use of RTP provides the link for easy combination with other multicast applications such as the well-known real-time services already explored by the MBone community. This is illustrated in the following section where we describe our approach to integrating a multicast audio capability into the shared virtual world. In the concluding section we note the status of our experimental implementation and describe plans for further development.

2. DISCUSSION

Over the last few years enthusiasm for a 3-DWeb viewer based on VRML has exploded on the Internet. A number of researchers have tackled the problem of developing support for sharing a VRML world. A good review of design considerations has been provided by Saar [13]. Most of the approaches that have been taken to extend VRML for multi-user support rely on simple network architectures based on the client-server model. Several research groups [2, 3, 4, 9] have taken a more ambitious approach introducing new network protocols and architectures that include basic multicast internet protocols rather than a centralized distribution scheme. While the proposal to exploit multicasting for shared virtual environments has been around for some time [9] a fully capable solution has not been agreed upon yet.

Our interest has arisen from a long-term research program in multicast tools for research collaboration, tele-learning and other shared collaboration environments [5, 7, 10]. Hence the focus is on interactions within a restricted space (e.g. conference room or building) with a small number of active players (probably less than 10). This is characteristic of a typical collaborative exercise. We have looked for an environment that can be rich in real-time interactive features, in particular audio, video, shared workboard, shared documents, making this a natural opportunity

* CRCC, Ottawa, K2H 8S2, Canada
EMail: {john.robinson, john.stewart, isabelle.labbe}@crc.ca

to exploit the multicast applications under development in the MBone community.

Active approaches include the work being done with VRML/DIS integration and the definition of VRTP [3]. Our approach meets some of the VRTP criteria such as the use of multicasting but is not based on DIS PDUs. We concur with the general observation that while DIS is very suitable for the special large-scale defense uses for which it was designed most DIS PDUs are not suitable for general-purpose virtual environments.

Other related work [2] includes the Distributed Worlds Transfer and communication Protocol (DWTP) where the emphasis is on reliability and scalability and the definition of a set of daemons to manage those features. We expect some of those ideas to become useful when we address more complex issues. However, noting for example that reliability is not important for position and orientation updates, we have taken a simple approach and concentrate instead on embedding shared virtual worlds into the interactive real-time multicast technology coming from the MBone research community.

Finally, the scheme proposed in [4] is closest in concept to our approach. Multicast IP is used for peer-to-peer communications among browser-clients and http is used, via a “Gatekeeper”, to locate and load the initial world.

An earlier project [6], concerning QoS management with the Internet Protocol version 6 (IPv6), developed a shared VRML viewer using FreeWRL [8] and the client-server VNet [15] to work on a Linux platform. The experimental implementation of audio enabled sharedVRML, built on that foundation, developing a multicast FreeWRL in combination with the multicast capable Robust Audio Tool (RAT).

Our FreeWRL viewer (Figure 1) can be launched via a Web browser however two factors have dissuaded us from using a browser-based interface at this stage of development: Java applets have restrictions concerning the generation of multicast packets; unresolved applet problems with EAI code on NT machines has led us to simplify and gain complete control of the development environment. Continuing with that simple approach, we have not developed a special MVIP control panel but rather use VRML objects for Shared Virtual World control. This allows us to move transparently to other control mechanisms, for instance, head-mounted displays or joystick control.

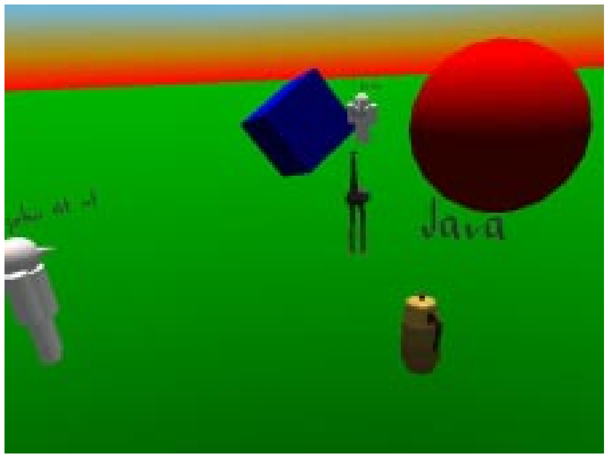

Figure 1: Screenshot of the Shared FreeWRL Viewer

3 THE MULTICAST VRML INTERCHANGE PROTOCOL (MVIP)

3.1 Architecture for Multicast VRML

A simple, compact VRML Interchange Protocol (VIP) that sends VRML commands over networks and allows multi-user participation has been described by Sonstein [15] and is being used by the VNet client-server pair in various distributed VRML systems. This provides a useful starting point from which the reliance on a server for host-host exchange can be removed and replaced with direct peer-to-peer communications using RTP, User Datagram Protocol (UDP) and multicast IP. We call this shared-multicast-virtual-world protocol Multicast VIP (MVIP). MVIP is implemented as a Java program that uses the services of RTP [14] and accesses VRML via the EAI (Figure 2).

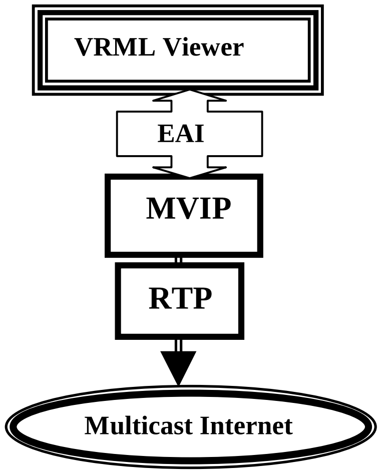

Figure 2:
The Interrelated Technologies Used for Multicast VRML.

Following VNet's example message types are defined for exchanging updates on position, orientation etc., an avatar insertion methodology and other information. An RTP packet type has been proposed [11] so that the suite of MVIP message types, modified from those defined for VIP, can be embedded into RTP packets for real-time exchange. Two classes of message type are defined so that time sensitive data flows in RTP while informational data flows in RTCP. These messages are generated from every participant (source) in the shared world and broadcast to all the other members of the multicast session.

When the viewer is launched, from a command line or the Session DiRectory (SDR), the world model is retrieved from an http repository. Informational message types are defined to retrieve, from an http repository, the specific avatar appearance selected by the other participants and, with a world-checksum, to maintain confidence (soft synchronization) in the common view of the shared world. The protocol layering for our shared virtual world is illustrated in Figure 3.

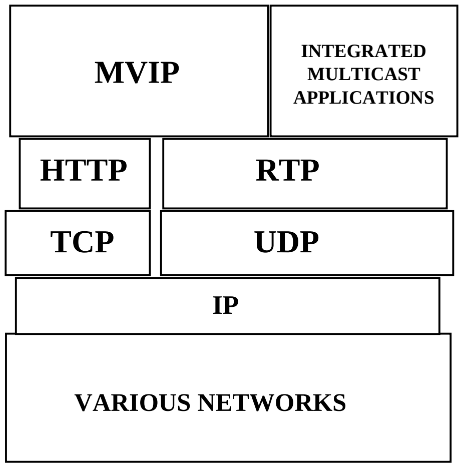

Figure 3: Protocol Layers for a Shared MVIP World

3.2 Message Types

When one moves in a shared world, the others sharing the world must see this movement, and they must know how to map an Avatar, and other services onto this movement. There are two classes of data that must be interchanged over a network to allow for this sharing:

a) Time-sensitive data
b) informational data

In MVIP the time sensitive data flows over RTP while the informational data flows over RTCP. The message types that have been defined for MVIP and the type IDs that have been assigned are given in Tables 1 & 2. The encoding column defines the bit pattern to be used in an implementation.

ID	Name	Encoding	Description
1	Position	float float float (32 bit)	x, y, z position in Cartesian coordinates; base coordinates and references supplied by the base world
2	Orientation	float float float float (32 bit)	3-vector of axis, angle (angle in radians) base coordinates and references supplied by the base world
3	Clicked	int int (32 bit unsigned)	SSRC of object that has been clicked; source of this action is given in RTP payload header
4	Interaction/Gestures	int utf8 (32 bit unsigned, string)	Avatar dependent action information
future	-	-	-

Table 1: Message Types and ID Assignment for Time-Sensitive Data

Time-sensitive data includes position/orientation changes that must be updated in a timely manner. These messages contain parameters (position and orientation with standard VRML meanings) that identify the current location (at time of packet generation) for that avatar.

Interaction between avatars can be seen by sending a "clicked" packet. The result of this clicked event is defined by either the avatars themselves, or by the world. For example, transferring ownership of a child-object (e.g. a book, a key, etc.) to a remote avatar may be accomplished by clicking on the destination.

An avatar can have gestures or other actions, defined and invoked by the owner of that avatar. Such gestures may include facial expressions (laugh, smile, frown, etc) or other actions, for example, morphing from one shape to another. By either including a VRML_ROUTE command or an external program, it is possible to have a clicked event initiate avatar gestures. An example of this may be a button on a display case, that when clicked, changes the items in the display. The format and functionality of the clicked and gesture message types is under development.

ID	**Name**	**Encoding**	**Description**
-1	Avatar Information	float float float utf8 (32 bit, string)	the scaling factor for an avatar, (along Cartesian coordinates) and the URL of an avatar to represent this SSRC[1].
-2	World Checksum	int (32 bit unsigned)	ensures that all players are playing in the same virtual world. (currently, no algorithm has been selected to generate this checksum.)
-3	Proximity Criteria	float float float (32 bit)	distributes the "personal space" around the avatar, for use in non-VRML collisions, audio closeness, and other[2].
-4	Amalgamation of tools information	int utf8 (32 bit unsigned, string)	This allows grouping of different media sources that may be independent on the RTCP CNAME parameter[3].
future	-	-	-

Table 2: Message Types and ID Assignment for Informational Data

[1] Message type: -1 (avatar information)
More than one avatar URL can be given. The first URL indicates the primary avatar, additional URLs indicate objects owned by this avatar. This information facilitates transfer of ownership via “clicked” packets.

[2] Message type: -3 (proximity criteria)
This message distributes information about the "personal space" around the avatar, for use in non-VRML collisions, audio closeness, and other. For example to determine audio closeness (i.e.who can hear me?) the results of the proximity detection calculations, as described in the next Section, could be transmitted.

[3] Message type: -4 (Amalgamation of tools information)
This allows grouping of different media sources that may be independent of the RTCP CNAME parameter. This grouping enables identification and correlation of such media streams on a configurable basis, for use by other participants. The owner of the resource is the SSRC in the originating RTP packet. Packets consist of groups of the following tuples: SSRC of tool (32 bit integer); MCast address/port (network dependent) (utf8 string).
This message type will be used when handling different multicast services for one avatar on more than one computer.

Informational data consists of the avatar presentation information (a reference to a VRML file), standard RTCP data such as a CNAME, and special items, for instance, a "world" checksum to ensure that all avatars are playing in the same world.
These are passed as APP fields in RTCP packets.

3.3 MVIProtocol in RTP

Packet types have been defined for the two classes of MVIP data. Time sensitive data that is carried in RTP packets and informational data that is carried as APP fields in RTCP packets. Figures 4 and 5 provide examples of MVIP messages embedded in RTP and RTCP packets respectively [14]. Definition of an RTP payload type to support this embedding has been proposed in the IETF AVT Working Group [11].

```
 0 1  2 3  4  5 6  7 8  9 0  1 2  3 4  5 6  7 8  9 0  1 2  3 4  5 6  7 8  9 0  1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  V |P|X|  CC=0  |M|        PT        |            sequence number          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp of initial frame                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  synchronization source (SSRC) identifier                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   MVIP_PT =1 |       Version       |                 undefined              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              X axis  (32 bit float)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Y axis  (32 bit float)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Z axis  (32 bit float)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: a Position update in RTP

Where:
V, P, X, CC, M, PT, sequence number, Timestamp, and SSRC are defined in the RTP specification [14].

MVIP_PT is the Shared VRML Packet Type (8 bits); this determines the remainder of the packet. Version (8 bits) is the version of MVIP being used. The SSRC identifier is used as the object identity.

```
 0 1  2 3  4  5 6  7 8  9 0  1 2  3 4  5 6  7 8  9 0  1 2  3 4  5 6  7 8  9 0  1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V  |P|      RC     |      PT=204        |                  length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                SSRC of sender                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  MVIP_PT = -1|       Version       |                 undefined              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     scaling factor on X axis  (32 bit float)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     scaling factor on  Y axis  (32 bit float)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     scaling factor on  Z axis  (32 bit float)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          URL of the avatar geometry                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        URL of the child 1-N (as needed)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Avatar information in RTCP

Where:
V, P, RC, PT, length, and SSRC are defined in the RTP specification [14]. (RTCP PT=204 identifies the application (APP) type).

MVIP_PT is the Shared VRML Packet Type (8 bits); this determines the remainder of the packet and Version (8 bits) is the version of MVIP.

4. AUDIO FOR SHARED VRML

Our goal was to create a realistic audio environment in our virtual world. For this we need audio proximity information to silence the audio from remote speakers and to amplify the volume as an avatar approaches. We also wanted stereo volume control to localize the direction of the audio source in 3D [1].

Our interest in embedding a multiparty interactive audio capability in the virtual world was aided with the newest release of the Robust Audio Tool (RAT v4) [12] from UCL. RAT v4 has the functionality to support audio proximity (already in previous versions) and stereo-based directionality (3D audio new in v4). The RAT application is launched at the same time as the VRML viewer with an announced or predetermined multicast address and port number. The MVIP program sends to RAT the login name chosen by the user. RAT uses this login name as its CNAME (RTP) for the session.

RAT will receive the audio signal from every transmitter in the multicast group. From these we want to distinguish those sources that are close enough to be heard. We have modified RAT by adding an “MVIP_filter” to the source code so that distant sources are discarded and the audio gain is set for loudness and direction based on information received from the MVIP program.

The MVIP program receives, via the EAI, the position/orientation of it’s avatar at all times as well as knowing about the position/orientation of other participants via the information it receives from the other participants in the position and orientation update messages. Using this information, the MVIP program performs periodic calculations to determine the proximity and orientation of the avatar relative to other avatars that are audio sources. (The mathematics for these calculations is given below). Demonstration showed that, for our prototype implementation, once per second was frequent enough for these calculations to give the subjective impression of real-time without the computation putting a heavy load on the processor. It remains for future study to determine how well this computationally complex problem will scale to many users

The MVIP program transmits the following information to RAT (Figure 6): name of each avatar that has been calculated to be within audio proximity; for each of those the associated audio gain and azimuth (for 3D audio). If no one is within proximity the string sent to RAT contains a null character.

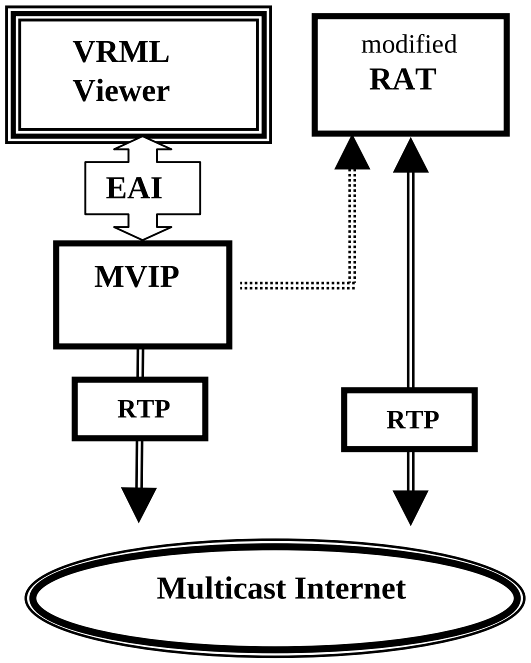

Figure 6: Integration of RAT into a Shared VRML World

RAT maintains a table with the names of all the avatars that are close enough and their associated audio output gain and azimuth. When RAT receives an update from the MVIP program the string is decoded and table updated with the new information.

When receiving an audio packet on the multicast address, RAT extracts the CNAME and determines from the table if the audio packet should be kept or discarded. (CNAME of the audio packet should match an avatar name in the table). For each of the avatars that is within proximity, the audio is rendered with the appropriate gain and 3D value.

Figure 7 shows the geometry that we have used for the location and orientation of two avatars relative to the origin of the virtual world. For simplicity, the elevation is not taken into account.

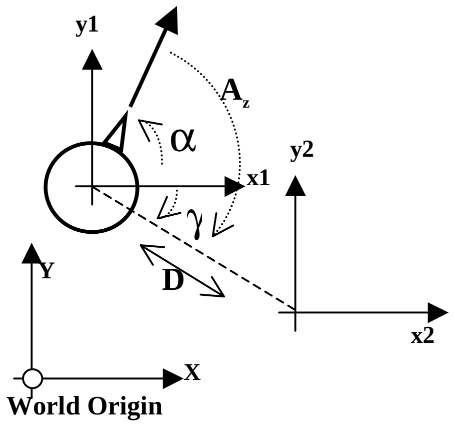

Figure 7: Geometry of the Virtual World as Seen from Above

To simplify computational complexity calculations are performed on the ground plane only. For each avatar (other than itself) in the shared virtual world the MVIP program performs the following steps:

- *a proximity detection calculation* i.e. computes the distance to the other avatars and verifies if that distance is within the threshold.

$$D < \text{Threshold}$$

where

$$D = (\Delta y^2 + \Delta x^2)^{1/2}$$

If the other avatar is within proximity, then the distance is equated (see below) to a RAT audio output gain/volume and the azimuth (A_z) is calculated.

- *an azimuth calculation* i.e. calculates the rotation of the avatar relative to the other avatar's position.

$$\gamma = \text{atan2}(\Delta y / \Delta x)$$

$$A_z = \alpha - \gamma$$

RAT uses the azimuth to balance the output to the right and left channels. For the calculation of the azimuth, the orientation of the receiver avatar (α) relative to the world origin is known from VRML.

In our prototype implementation, 10 meters was selected as the threshold, i.e. if the distance is greater than 10 m, it is declared that the other avatar was too far away to be heard. If it is calculated that the other avatar is within proximity, then a linear function is assumed for the mapping of the distance to RAT audio output gain. Specifically:

$$\text{Gain} = 100 - 10D$$

where D is the distance, as above.

5. STATUS AND FUTURE DIRECTIONS

To test the ideas presented above an existing multi-user VRML (FreeWRL/VNet) is being converted into a multicast shared virtual world. In an earlier project [6] a shared VRML viewer using FreeWRL [8] and the client-server VNet [15] was developed on a Linux platform. Our experimental implementation of audio enabled sharedVRML has built on that foundation, replacing VNet with MVIP to give a multicast FreeWRL in combination with an MBone audio tool (RAT) [12]. User response to this new capability has been very positive. A simple approach has been taken that concentrates on embedding shared virtual worlds into the interactive real-time multicast technology (IP multicast, RTP, etc.) coming from the MBone research community.

While the RTP embedding strategy has been illustrated with MVIP it was designed to be independent of any particular distributed simulation transport protocol and can be used with any open source VRML browser. It remains for future study to explore this approach for example with VRTP and DIS.

More complex issues arising from reliability and scalability will be important in future work. Noting, for example that reliability is not important for position and orientation updates and that applications in research collaboration, our initial interest, are likely to involve only a small number of participants, those issues have been downplayed initially. This work has built on an earlier project that included a study of QoS management in IPv6 to support distributed simulation. It remains for future investigation to determine if that can provide sufficient reliability as an alternative to other approaches, such as reliable multicast.

ACKNOWLEDGEMENT

The authors wish to thank Colin Perkins, Orion Hodson and other members of the RAT team at University College London for their help with RAT. We also thank Nicolas Georganas and the members of the DIVE team at University of Ottawa for their ideas during the CANARIE/DIVE project

REFERENCES

[1] J.Bolot & S.Fosse-Parisis. Adding Voice to Distributed Games on the Internet, In *IEEE Infocom'98*, pages 480-487. San Francisco 1998.

[2] W. Broll. DWTP - An Internet Protocol for Shared Virtual Environments. In *Proceedings of the VRML'98 Symposium*, Monterey California, February 1998.

[3] D.Brutzman, M.Zyda, K.Watsen and M.Macedonia. Virtual Reality Transfer Protocol (vrtp) Design Rationale, In *Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises: Sharing a Distributed Virtual Reality*, pages 179-186. MIT, Cambridge, Massachusetts. June 1997

[4] J.A.Carson & A.F.Clark. Multicast Shared Virtual Worlds using VRML97, In *Proceedings of VRML'99*, pages 133-140, 1999.

[5] Collaborative Virtual Workspace (CVW), http://www.mitre.org/pubs/showcase/cvw.html

[6] Distributed Interactive Virtual Environment (DIVE) over CA*net II: QoS Management, IPv6 and Performance Tools. http://www.mcrlab.uottawa.ca/research/DIVE.htm.

[7] D.W.Fellner and A.Hopp. VR-Lab - A Distributed Multi-User Environment for Educational Purposes and Presentations. In *Proceedings of VRML'99*, paes 121-131. 1999.

[8] FreeWRL – An Open Source VRML Browser for Linux. http://www.crc.ca/FreeWRL/.

[9] M.Macedonia, M.Zyda, D.Pratt et al. Exploiting Reality with Multicast Groups: A Network Architecture for Large-Scale Virtual Environments. In *Proceedings of IEEE VRAIS'95*, pages 2-10. March 1995.

[10] MECCANO: Multimedia Education and Conferencing Collaboration over ATM Networks and Others, http://www-mice.cs.ucl.ac.uk/multimedia/projects/meccano/.

[11] J.Robinson and J.Stewart. RTP Payload format for Shared Multicast Virtual Worlds (SMVW). work in progress - draft-stewart-avt-00.txt, 25 June 1999.

[12] Robust Audio Tool (RAT). http://www-mice.cs.ucl.ac.uk/multimedia/software/rat.

[13] K.Saar. VIRTUS: A Collaborative Multi-User Platform. In *Proceedings of VRML'99*, pages 141-152. 1999.

[14] H.Schulzrinne, S.Casner, R.Frederick and V.Jacobson. RTP: A Transport Protocol for Real-Time Applications, RFC 1889, IETF AVTWG, January 1996.

[15] draft-ietf-avt-rtp-new-005.txt, October 1999.

[16] The VRML Interchange Protocol: VNet. http://ariadne.iz.net/~jeffs/vnet/.

[17] Web3D Consortium DIS-Java-VRML Working Group. http://www.web3d.org/WorkingGroups/vrtp/dis-Java-vrml.

[18] M.Zyda and S.Singhal. Networked Virtual Environments: Design and Implementation. Addison Wesley Publ., August 1999.