Group Management and Quality of Service Adaptation in Distributed Virtual Environments

Nat Pryce

Department of Computing,
Imperial College,
180 Queen's Gate
London SW7 2BZ

Tel: *0171-594 8394*
Fax: *0171 581 8024*
e-mail: *np2@doc.ic.ac.uk*

Abstract: This paper proposes a protocol that manages groups in a distributed virtual environment and adapts to changing quality of service levels. The protocol makes use of application-level information about the relative positions of entities in the virtual space to detect clusters of entities. Since entities in close proximity are more likely to interact, groups are dynamically allocated to clusters as they form, thus allocating network resources to those end systems which require them, and reclaimed as the members of those clusters move apart. Clusters are organised as a tree structure – large clusters are subdivided into smaller clusters – and each node in the tree has an associated group. This hierarchy is a form layered encoding scheme and is used by participants to adapt to congestion caused by other users of the network

Keywords: distributed virtual environments, multicast protocols, quality of service, adaptation

1 Introduction

Distributed virtual environments (DVEs) promise many advantages over both current single-user virtual reality systems and computer supported co-operative work (CSCW) applications. Distributed virtual environments allow multiple users to share the same environment and so form a shared workspace in which the explicit embodiment of users allows the intrusive “floor-control” mechanisms used by desktop CSCW tools to be replaces by a more natural use of space and body language. DVEs also allow remote physical processes to be visualised and controlled and so provide a way of sharing expensive resources. For instance, a DVE system could allow scientists at one site perform experiments on expensive equipment only available at another. In addition, distributing the computation required to implement a virtual environment promises performance improvements due to parallelism.

However, a protocol that can be used by distributed virtual environment applications to communicate over wide area networks, such as the Internet, must make efficient use of network resources and adapt to changes in quality of service caused by other users of the network. This paper examines existing protocols for distributed virtual environments and proposes a protocol that will address their shortcomings in these respects.

2 Distributed Virtual Environments

We take a broad view of the term “virtual environment”, using the term to mean any interactive application in which information is presented spatially within some common co-ordinate system. Virtual environments need not be three-dimensional; examples of two-dimensional distributed virtual environment applications include shared whiteboard applications and the Kansas distributed teaching system (Sun, 1997). A DVE system can be considered from two viewpoints: the perceptual viewpoint of what is in the environment and the implementation viewpoint of how the “illusion” of that environment is maintained.

2.1 Perceptual Viewpoint

Perceptually, a DVE is made up of any number of disjoint *spaces*, also known as *worlds*, which contain a number of media *entities*, such as graphical shapes or sound sources. Each world has an associated set of *laws* which govern the behaviour and interactions of the entities in that world; entities can interact through collisions and forces for instance. The worlds can be totally separate or can be hyperlinked, much like hyperlinked documents on the World-Wide Web; hyperlinks between worlds are often known as *portals*.

2.2 Implementation Viewpoint

From an implementation viewpoint, each world is shared by a number of *agents* which communicate via some shared communication medium, according to a predefined DVE *protocol*, in order to maintain a consensual model of the entities in the world. It is important to note that this implementation view is inherently *group oriented*: a group of agents interacts on a peer-to-peer level with no single agent acting as a server or co-ordinator.

For a DVE to have any useful application, some agents must interface to real-world processes in order to create visualisations of these processes or to control these processes in response to events occurring in the virtual environment. An important example of such an agent is the interface between a user and the virtual world which creates a visualisation of the user’s position and orientation in the world and renders a view of the world onto the user’s display.

Agents hide their implementation behind interfaces through which they communicate using the DVE protocol. Agents themselves can be concurrent, distributed or parallel programs but such details are not of interest to the protocol. Instead this paper will concentrate on the protocol itself and on the services which are necessary to support that protocol.

3 Distributed Environment Protocols

Distributed virtual environment applications have features which make them significantly different from traditional distributed computing applications, such as the World Wide Web, and current multimedia streaming applications, such as audio/video conferencing. Unlike traditional distributed systems, DVEs require real-time communication of multiple forms of media that must be synchronised before presentation to the user. The real-time nature of DVEs means that, like multimedia streaming applications, they are sensitive to changes in the levels of quality of service provided by the underlying network.

However, unlike media streaming applications, each agent in a DVE system maintains a significant amount of state about the current entities in the environment. Certain changes to this state, such as the introduction and removal of entities, require reliable transmission while others, such as the periodic update of an individual entity, can be treated in much the same way as a frame of audio or video. A further difference between distributed virtual environments and multimedia streaming applications is that multimedia applications typically stream samples in one direction – from the sender to eventual display at the receiver – and provide little support for interaction between the receiver and the sender. In contrast, DVEs are group structured, with all parties both sending and receiving data, and highly interactive requiring groups to be created and discarded as users interact in the virtual space.

Because DVE applications are group oriented, the most efficient way of implementing them is to use multicast group protocols provided by the network layer. The creation of a multicast group reserves scarce network resources, such as table entries in routers. A DVE protocol must manage groups effectively so that network resources are used to support volumes of virtual space in which interaction is taking place and reclaimed when that interaction ends. In addition to managing groups, a DVE protocol must manage bandwidth utilisation to avoid congestion in network routers and at end systems. Congestion causes packets to be queued or discarded at routers, thereby increasing the latency and jitter apparent to the end system. Because the network is not usually dedicated to any single DVE application, the protocol must also be able to detect congestion and quality-of-service degradation caused by other applications using network, and adapt its use of the network in a way which is compatible with other congestion control schemes.

Most early DVE systems did not use multicast protocols for communication, instead relying on point-to-point connections between peer agents (Snowdon, 1994; Greenhalgh, 1995) or between multiple client agents and a central server. Those that use multicast protocols take different approaches to the allocation and management of multicast groups.

3.1 SIMNET and DIS

The SIMNET (Calvin, 1993) and DIS (IEEE, 1993) protocols, used for vehicle-level military simulations running over dedicated networks and internetworks, uses broadcast communication between agents. The state of each entity is broadcast over the network by the agent simulating that entity and received by remote agents so that they can track the state of the entity. A "dead-reckoning" algorithm is employed to both reduce the bandwidth requirements of the simulation and to decouple the simulation rates of individual agents.

The use of broadcast improves reliability, because no agent relies on state stored at any single node, and removes any bottlenecks which would be caused by a single server.
However, the stateless, decentralised architecture has a number of disadvantages. The lack of a centralised server or reliable group protocols means that entities must transmit their entire state in each update message so that nodes which joined the simulation late can recreate the entire state of the simulation; this means that update messages often include information which is constant or of no probable interest to many nodes, such as aircraft markings, and that new nodes must wait for several seconds before joining the simulation to ensure that they have received all possible update messages. The stateless nature of the protocol also means that entities which are static, such as stationary or destroyed vehicles, must periodically broadcast update messages; such "keep-alive" messages can comprise 70% of all network traffic in a large simulation. In addition, the use of broadcast means that all update messages are received by all nodes, whether they are relevant to them or not and limits DIS to use on private internets.

3.2 NPSNET

NPSNET-IV (Macedonia, 1994) extends the original DIS architecture to address many of these problems. NPSNET uses IP Multicast (Tannenbaum, 1996), so that it can be used over general internets, and manages the multicast groups that it uses in order to reduce unnecessary network traffic (Macedonia, 1995). NPSNET divides the terrain of the simulation into hexagonal cells, each of which corresponds to a single multicast group. Entities only transmit update messages to the group of their current cell, but listen for updates from their current cell and surrounding cells. The diagram in Figure 1 shows how an entity's area of interest maps onto multicast groups: the entity sends updates to the group corresponding to the darkly shaded cell and listens for updates in all shaded cells; the heavily outlined cells show which cells must be added and removed from the listening set as the entity moves between cells.

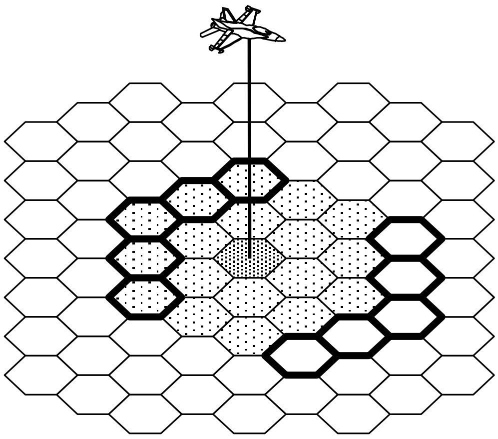

Figure 1. Allocation of Multicast Groups in NPSNET-IV

The entity that has been in a cell the longest is defined to be the "group leader". When an entity joins the cell, it asks the group leader for the current state of the group; the group leader sends this information using a point-to-point connection, such as a TCP stream, to avoid sending unnecessary traffic to all members of the group. This approach reduces the amount of information required in each update message and removes the need for keep-alive messages. It also reduces the time taken for a new member to join the simulation from several seconds to a few milliseconds. The authors state that sending the current state to new group members requires little processing effort compared to normal simulation and so doesn't noticeably degrade the performance of the group leader. The time taken for entities to cross a cell is typically in the magnitude of hours and the oldest member of the cell is often a static entity, such as a bridge, which has little simulation processing to perform anyway.

NPSNET solves many of the problems inherent in the DIS protocols, but is still most suitable for terrain-based vehicle simulations. For three-dimensional applications, such as data visualisation, the two-dimensional division of the simulation into cells might not be appropriate and the fixed allocation of IP-Multicast groups to cells makes inefficient use of network resources and does not take the capabilities of the underlying network links into account when allocating resources.

3.3 MASSIVE

MASSIVE (Greenhalgh, 1995) is a DVE system designed for teleconferencing. The first version of MASSIVE used a hybrid client-server and peer-to-peer architecture. Each agent advertised an "aura" about itself that indicated the volume of space in which it was interested. Agents transmitted update information to a server that set up point-to-point links between entities when their auras intersected. Agents connected in this way communicated directly with one another and the server until the server detected that their auras no longer intersected, at which point it tore down the point-to-point connection between them.

Originally, MASSIVE used a reliable connection oriented protocol to carry messages between agents. However, this resulted in poor real-time performance and high bandwidth usage that scaled poorly with large numbers of inhabitants of the virtual world. To address these issues, MASSIVE now uses a multicast protocol (Greenhalgh, 1996) and represents multicast groups as entities within the environment. For example, a meeting room could be modelled as an entity made up of walls, windows, a door and a multicast group that is used by entities within the room to communicate amongst themselves. Entities that contain groups can be controlled in the same way as any other entity, and so can be made to follow crowds, for example, to provide higher bandwidth for communication between crowd members. MASSIVE also allows group entities to act as "aggregates", hiding the members of the group from the rest of the virtual world and representing them in a simpler form. These mechanisms reduce bandwidth requirements and improve scalability. However, they only postpone the problems that are caused by the use of single group per world; for example, it would be possible for a so many entities to enter a meeting room that the combined traffic overloaded the bandwidth available for group assigned to that room. Additionally, the use of dynamic groups involves writing application-specific agents and increases the complexity of the programmer's task.

3.4 DIVE

The DIVE protocol (Carlsson, 1995) was designed to support interactive conferencing and visualisation applications. Each virtual world is stored as a database that is shared between agents using a reliable groupcast protocol (Birman, 1993). The use of reliable groupcast reduces the interactivity of the system and has large bandwidth requirements. These drawbacks, combined with the use of a single multicast group per world reduces the scalability of the system. To address these problems, DIVE now uses a light-weight groupcast protocol and multiple multicast groups per world, organised in similar manner to those in MASSIVE.

3.5 Summary

No existing DVE protocol manages groups in an adequate manner. DIS and NPSNET use a static allocation of groups to volumes of virtual space. MASSIVE and DIVE allow a more dynamic approach to group allocation by assigning groups to entities within the world. However, groups must be assigned to volumes of space "by hand" and mobile groups require requires significant programming effort. Furthermore, no current DVE protocols provide support for quality of service adaptation.

We argue that an efficient use of network resources requires a group management protocol that is dynamic and adapts to both the distributed state of the application and the current state of the network over which that application is running. This kind of group management is complex and should be hidden from the application programmer as much as possible. We are currently working on a group management protocol for distributed virtual environment applications that embodies these principles. The following sections describe our approach and how we envisage that our protocol will fulfil the requirements.

4 Group Management for Distributed Virtual Environments

A multicast group provides a single address with which any machine on the network can address a group of machines without needing to know the exact membership of the group. The network provides this facility by storing routing information and reserving bandwidth for each group at appropriate routers in the network and communicating this information between routers as group members join and leave the group. Multicast groups therefore consume valuable memory, bandwidth and address-space resources within the network and so must be carefully allocated and reclaimed in order not to overload the capabilities of the network. Our approach to group management dynamically allocates a hierarchy of multicast groups to volumes of space in which interaction is taking place and uses this hierarchy to support quality of service adaptation.

4.1 Group Management

Multicast groups are allocated to volumes of space. Initially, all entities are members of a single group that encompasses the entire virtual world. A single group is not adequate to support a virtual world containing many entities, so new groups must be allocated to support interactions between entities. This is performed dynamically when entities detect that the quality of service available to the group has degraded. An entity that detects a QoS degradation transmits a request to the other members of the group to start the sub-division process. If enough members determine that their QoS is inadequate, they co-operate to detect spatial clusters of entities and allocate new groups to the sub-volumes of space containing those entities. The entities then become members of the group allocated to the volume of space in which they are located.

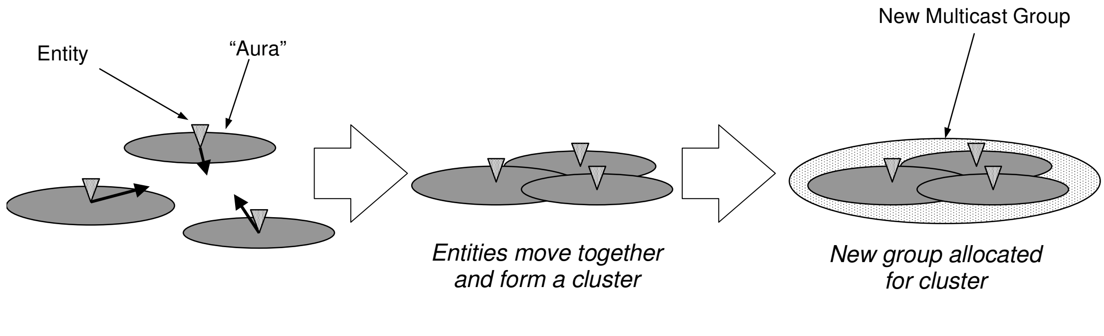

Figure 2. Groups are allocated to clusters of entities

This process is continued recursively, allocating further groups to smaller clusters of entities within the new group if QoS continues to degrade. Figure 3 shows how this protocol would recursively divide a two dimensional world into a tree of spatially enclosed groups. Each entity is a member of all the groups that encloses it and receives messages from all enclosing groups. However, an entity transmits each of its update messages to only one of the groups which encloses it. It transmits updates most frequently to the innermost group, less frequently to the next group outwards, less frequently still to the group enclosing that, and so on.

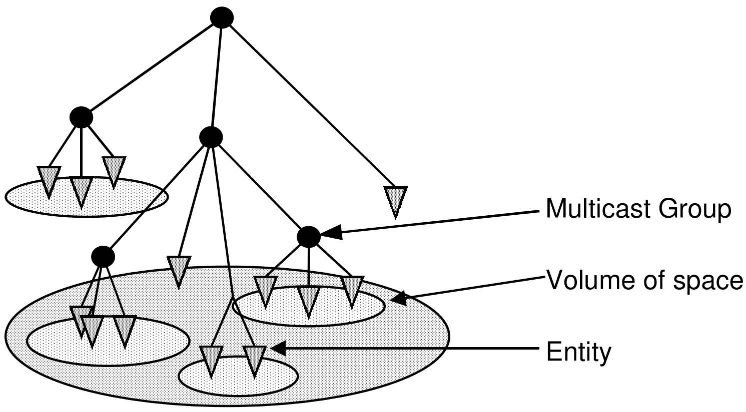

Figure 3. Spatial and hierarchical nesting of groups around clusters of entities

Thus each entity is provided with high-fidelity information about the closest other entities but still retains a peripheral, low-fidelity awareness of other entities in the world. Since an entity is most likely to interact with entities that are closest to it, we believe this trade off will be suitable for most applications.

4.2 Quality of Service Adaptation

If an entity detects a QoS degradation but other members of the group do not need to subdivide the group, the entity must adjust its own use of network to adapt to the new QoS level. The hierarchical organisation of spatial groups facilitates QoS adaptation for individual group members because it implements a form of layered media encoding (Shacham, 1992). Layered encoding, also known as sub-band encoding, is a scheme used by media stream protocols to implement QoS adaptation. Media data is transmitted as a number of separate streams which are integrated by the receiver before presentation. Each stream has a preassigned priority with the highest priority stream carrying a low fidelity representation of the media and each further stream carrying data that can be combined with all higher priority streams improve the final fidelity. As the network becomes overloaded, data is discarded from low-priority streams before it is discarded from high priority streams. Receivers can therefore continue to present the media smoothly, although the fidelity of the presentation will be lowered.

In the case of the virtual environment protocol outlined in section 4.1, the highest priority layer is the group that encompasses the entire world while the lowest priority layer for any entity is the innermost group around that entities location. Each layer increases the temporal fidelity of the information transmitted for any entity, since messages are transmitted less often in higher priority groups and more often in lower priority groups. A loss in temporal fidelity will typically result in the loss of spatial fidelity at the receiver because less accurate information is available as a basis for dead reckoning.

The use of a layered encoding means that the Receiver-driven Layered Multicast (RLM) protocol (McCanne, 1996) can be adapted for use by distributed virtual environment applications. RLM was designed for the transmission of audio and video over large IP internets and follows the principles of *Integrated Layer Processing* (Clark, 1990) by integrating application level information with assumptions about the underlying network architecture to make the most effective use of network resources.

The RLM protocol takes advantage of the typical architecture of an wide area internet in which local area networks (LANs) providing high bandwidth communication are connected to a high bandwidth backbone by low capacity links. Congestion occurs when too many messages are routed over the low capacity link, resulting in queuing delays and packet loss at the routers at the end of the link.

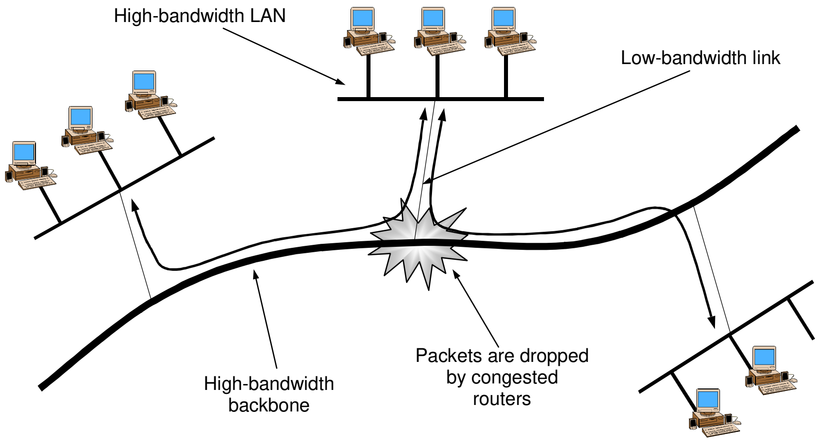

Figure 4. Typical internet architecture

When a host on a LAN joins a group, routers at the end of the link start sending messages transmitted to that group across the link and over the LAN. Similarly, when all hosts on the LAN that had joined toe group have dropped their membership, the router stops sending messages from that group across the link.

When QoS degradation is detected by a group member on a LAN, it drops membership of its lowest priority group. All other members of the group on the same LAN will also detect QoS degradation and drop membership of the group, so stopping transmission of the group's traffic over the low-bandwidth link. If congestion is still detected, further groups will be dropped, until congestion no longer occurs or no more groups can be dropped. In the latter case, the user must be alerted so that they can take appropriate action, such as switching to a higher cost service or terminating the application.

Once bandwidth usage has been reduced to adapt to network congestion, the RLM protocol performs tests to detect whether more bandwidth has become available. After a random delay, a host rejoins the next group of lower priority than its current group and analyses the available QoS to determine if is acceptable. To stop more than one host on the same LAN attempting to join the group, so causing more congestion, the first host to perform the join experiment broadcasts an announcement that it is rejoining the group over the LAN and also broadcasts the result of the join experiment – whether the QoS is acceptable or not – when the experiment is complete. When a host receives an announcement of a join experiment, it does not attempt to join the same group, but waits to receive the result of the experiment.

5 Current Status and Further Work

The protocol described in this paper is currently being designed and implemented on Windows NT using the Regent framework (Pryce, 1998), an object-oriented framework for building distributed systems and communication protocol software. Currently, quadtrees/octrees are used to detect clusters of entities. This algorithm has some undesirable properties and we are investigating other techniques. We intend to perform tests to demonstrate that this protocol does reduce the bandwidth requirements of a distributed virtual environment application and that the QoS adaptation protocol is compatible with the end-to-end congestion avoidance mechanisms used by "elastic" protocols such as TCP/IP.

References

Benford, S. D., Bullock, A. N., Cook, N. L., Harvey, P., Ingram, R. J. and Lee, O., *From Rooms to Cyberspace: Models of Interaction in Large Virtual Computer Spaces*. Interacting With Computers, 5 (2), pp 217-237, Butterworth-Heinmann, 1993.

Birman, K. *The Process Group Approach to Reliable Distributed Computing*. Communications of the ACM, December, 1993, 37-53.

Calvin, J., Dickens, A., Gaines, R., Metzger, P., Miller, D. and Owen, D. *The SIMNET Virtual World Architecture*. IEEE VRAIS, September 18-22 1993, Seattle, Washington.

Carlsson and Hagsand. *DIVE - A Multi User Virtual Reality System*. IEEE VRAIS, September 18-22 1993, Seattle, Washington.

Clark, D.D., and Tennenhouse, D.L. *Architectural considerations for a new generation of protocols*, in Proc. SIGCOMM'90, Philadelphia, PA, September 1990, ACM.

Greenhalgh, C. M. and Benford, S. D., *MASSIVE: a Distributed Virtual Reality System Incorporating Spatial Trading*, Proc. 15th IEEE International Conference on Distributed Computing Systems (ICDCS'95), Vancouver, Canada, May 30-June 2, 1995, IEEE Computer Society, pp 27-34.

Greenhalgh, C. M. *Dynamic, Embodied Multicast Groups in MASSIVE-2*. Technical Report NOTTCS-TR-96-8, Department of Computer Science, The University of Nottingham, UK, 1996.

Institute of Electrical and Electronics Engineers, International Standard, ANSI/IEEE Std 1278-1993, *Standard for Information Technology, Protocols for Distributed Interactive Simulation*. March 1993.

Macedonia, M.R., Zyda, M.J., Pratt, D.R., Barham, P.T., and Zeswitz, S. *NPSNET: A Network Software Architecture for Large Scale Virtual Environments*. MIT Presence 3(4), 1994.

Macedonia, M.R., Zyda, M.J., Pratt, D.R., Brutzman, D.P. and Barham, P.T. *Exploiting Reality with Multicast Groups*, IEEE Computer Graphics & Applications, September 1995, pp.38-45.

McCanne, S., Jacobson, V. and Vetterli, M. *Receiver-driven Layered Multicast*. ACM SIGCOMM'96, August 1996, Stanford, CA.

Pryce, N., and Crane, S. *A Model of Interaction in Concurrent and Distributed Systems*. Submitted to the Second International Workshop on Development and Evolution of Software Architectures for Product Families, Las Palmas de Gran Canaria, Spain, February 26-27 1998.

Shacham, N. *Multipoint Communication by Hierarchically Encoded Data*. In Proc. Infocomm'92, pp. 2107-2114, Florence, Italy, May 1992.

Snowdon, D., and West, A.J. *The AVIARY VR System: A Prototype Implementation*. 6th ERCIM Workshop, June 1994, Stockholm.

Sun Microsystems Laboratories. *Kansas: a Distributed Classroom*. Electronic document. 1997. <URL: http://www.sunlabs.com/research/distancelearning/kansas.html>

Tannenbaum, A.S. *Computer Networks, Third Edition*. Published by Prentice Hall, 1996. ISBN 0-13-394248-1.