1995-09

# Exploiting Reality with Multicast Groups

Macedonia, Michael R.

IEEE Computer Graphics and Applications, September 1995

http://hdl.handle.net/10945/40168

# Exploiting Reality with Multicast Groups

Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Donald P. Brutzman, and Paul T. Barham
*Naval Postgraduate School*

**Large, multiuser virtual environments demand high-bandwidth communications, in part to keep each entity aware of others' actions. Limiting an entity to its area of interest permits larger environments.**

Multiuser networked virtual worlds have generated intense interest in the graphics community. High-bandwidth wide area communications, the success of World Wide Web applications such as the National Center for Supercomputing Application's Mosaic browser, and government funding of Distributed Interactive Simulation (DIS) have fueled the desire to expand networked virtual worlds beyond local area networks. However, the Internet has proved a challenging environment for real-time applications such as interactive virtual worlds and multimedia.

We wanted to expand the capabilities of simulations and virtual environments (VEs) by exploiting multicast networks to serve 1,000 or more simultaneous users. Today, military simulations demand these numbers, and in the future, distributed interactive entertainment applications—such as multiplayer games—will require scalable network architectures to provide rich environments and profitable returns.

This article describes our investigation into developing large distributed simulations. The architecture we describe logically partitions virtual environments by associating spatial, temporal, and functional classes with network multicast groups. We exploit the actual characteristics of the real-world large-scale environments being simulated by focusing or restricting an entity's processing and network resources to its area of interest via a local area-of-interest manager (AOIM). For example, a simulated infantryman in a virtual environment doesn't need to know the condition of a simulated truck 20 virtual kilometers away.

## Problems scaling DIS

Advances in computer architectures and graphics, as well as standards such as the IEEE 1278 Distributed Interactive Simulation (DIS) and BBN (Bolt Beranek and Newman) Simnet protocols have made small-scale (less than 300 interactive players), realistic, man-in-the-loop simulations possible.[1] The military has used these standards for several years. Unfortunately, Simnet, which was developed for small-unit training, and its descendant, DIS, are unsuitable for large-scale multiplayer virtual environments.

Several major problems associated with scaling the current suite of DIS protocols illustrate the difficulty of building large virtual environments:

■ *Enormous bandwidth and computational requirements for large-scale simulation.* A Simnet or DIS simulation with 100,000 players can require hundreds of megabits per second of network bandwidth to each computer in the simulation—too expensive today.

Maintaining the state of all other entities, particularly with dead-reckoning algorithms (which use second-order kinematics equations), will be a major bottleneck for large-scale simulation. Recent experiences with the US Army's Synthetic Theater of War (STOW) exercises confirmed this.[2]

Faster computers and networks will not necessarily satisfy these needs. First, faster networks require faster processors merely to copy packets from the network into user space even before the application touches the protocol data unit (PDU). Second, the creeping demand for more realism (for example, collision detection and constraint satisfaction) will introduce a rapid rise in computational and spatial complexity with even modest-size virtual environments.[3]

We conjecture that a single host can realistically manage only about 1,000 entities in real time, despite future advances in computer graphics architectures, due to computational complexity.

■ *Multiplexing of different media at the application layer.* The current DIS protocol requires the application to multiplex and demultiplex different types of real-time data (for example, simulation packets, audio, and video) at the application layer rather than at the network or transport layers. Therefore, the virtual environment must treat continuous video streams the same as bursty simulation traffic—that is, through allocation of buffers and timing at the application layer. This can be very expensive.

0272-17-16/95/$4.00 © 1995 IEEE

■ *Lack of an efficient method for handling static objects.* Events like explosions can change many static entities, such as bridges and buildings. These and other stationary objects must send update messages at regular intervals to inform participants of their current state. For example, a tank that has been destroyed must constantly notify the environment that it is dead, in order to inform new entrants or other entities that might have missed the original state-change message.

■ *Models and world databases must be replicated at each simulator.* DIS has no mechanism to distribute objects on demand. This is a necessity for large simulations, particularly when simulators are heterogeneous and controlled by different organizations, among whom minimal coordination is expected prior to an exercise.

Furthermore, it is neither feasible nor efficient for each simulator to store every model and database for a 100,000-entity simulation. For example, a human simulation (say, a dismounted infantryman) on land normally need not concern itself with naval vessels—unless the infantryman is near enough to the ocean to see them.

The DIS protocol's origins explain some of these shortcomings:

■ *Event and state paradigm.* DIS requires VE simulations to be stateless; data is fully distributed among the participating hosts, and entities are semi-persistent. Therefore, every entity must learn about every event, just on the chance that it might need to know it. To accomplish this, the protocol asks each entity to regularly communicate its state information (as an entity state protocol data unit, or ESPDU) to every participant—even though the ESPDU data may be redundant or unnecessary (such as aircraft markings). These *keep-alive* ESPDU messages can comprise 70 percent of the traffic in large simulations.

In the real world, however, different entities have different sensing capabilities, and these can be translated into a virtual entity's data requirements. In a large virtual environment, two ground vehicles 200 kilometers apart probably don't need to know of each other. Yet, under the current architecture, they must inform each other of state changes and updates. This avoids the reliability problems of a central server, simplifies communication protocols, and minimizes latency while guaranteeing that hosts entering a simulation would eventually build their entity database through entity state and event messages. Furthermore, broadcast ESPDU updates help maintain a consistent view among simulators, within a particular tolerance.

■ *Real-time system trade-offs.* Large distributed systems often compromise reliability (that is, a guarantee that sent data has been received) to achieve real-time performance. Truly reliable systems require acknowledgment schemes, such as the one used in Transport Control Protocol (TCP). These defeat the notion of real time, particularly if a player host must establish a virtual connection with every other entity host to ensure that they received data correctly. Thus large environments must rely on connectionless (and therefore unreliable) network protocols, such as the User Datagram Protocol (UDP), for wide-area communications.

## Related Work

Partitioning virtual worlds into spaces is a common metaphor for virtual environments. Multi-user dungeons (MUDs) have used this idea, and projects like Jupiter from Xerox PARC (Palo Alto Research Center) have extended it to associating *rooms* with multicast video and audio teleconferences.[1] Benford described a concept for the spatial interaction of objects in a large-scale virtual environment.[2] Benford's spatial model uses different levels of awareness between objects based on their relative distance and mediated through a negotiation mechanism.

An implementation using DIVE (Distributed Interactive Virtual Environment) uses "standard VR collision detection" to determine when transitions between awareness levels should occur.[3]

The MASSIVE (Model, Architecture, and System for Spatial Interaction in Virtual Environments) from the University of Nottingham project also uses this approach. However, the need for collision detection, reliable communication, and strong data consistency have made it difficult for DIVE and MASSIVE to scale beyond a handful of users.[2] This may change as their developers pursue multicast communications and weaker data consistency.

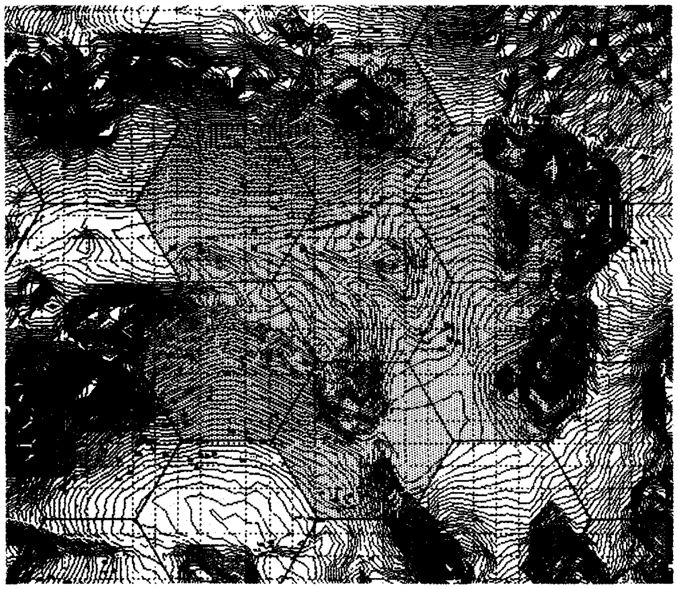
### References

1. P. Curtis and D.A. Nichols, "MUDs Grow Up: Social Virtual Reality in the Real World," *Proc. IEEE Computer Conf.*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 193-200 (or ftp.parc.xerox.com/pub/MOO/papers/MUDsGrowUp.ps).
2. S. Benford et al., "Managing Mutual Awareness in Collaborative Virtual Environments," *Proc. Virtual Reality Software and Technology*, ACM Press, New York, 1994, pp. 223-236.
3. C. Carlsson and O. Hagsand, "DIVE—A Multi-User Virtual Reality System," *Proc. Virtual Reality Ann. Int'l Symp.*, IEEE Press, Piscataway, N.J., 1993, pp. 394-400.

Real-time environments need to avoid transactions between individual entities, since they require reliable communications. Furthermore, schemes that use a central database do not work well in a large virtual environment due to I/O contention. For example, AT&T's Imagination network limits the number of concurrent players in a game to four because they are centrally served and bandwidth is limited to modem speed (less than 28 Kbps).

■ *No middleware layer.* DIS does not mediate between distributed VE applications and the network. It uses a bridged network in which every message is broadcast to every entity.

However, large simulations require internetworking (routing over the network layer) because this allows the use of commercial services rather than private networks. This lets us connect diverse, remote sites; use different local network topologies and technologies (such as Ethernet and FDDI); and take advantage of rich topologies for partitioning bandwidth—which provides robustness and optimizes routes for minimizing latency. Since current standards confine DIS to the data link layer, we limit ourselves to using bridges, which are an order of magnitude slower than routers when reconfiguring after topological changes. Bridges also limit the number of stations to the tens of thousands. A router



1 An armored brigade in a combat-training exercise at the US Army National Training Center, Fort Irwin, California. The area is 60 by 50 kilometers. Friendly vehicles are blue; enemy vehicles are red.

network is limited only by the numbers accommodated in the address space.

■ *Origins as small training-unit systems for LANs.* Many of these problems devolve from the fact that until recently DIS and Simnet were used exclusively for small-scale training simulations. In this mode it has been relatively easy to ensure that the virtual environment components have homogenous sets of models and terrain databases by replicating them at each host. The monolithic nature of these small-scale environments, which could be distributed over a local area network, obviated the need for middleware. Hence, broadcast communication sufficed for these limited environments.

These origins also influence current assumptions about the density and activity rates of entities in large simulations that do not necessarily match the real world. Simnet players participated for short periods (several hours) and were very active because simulation aimed to train crews in coordinated drills. Furthermore, entities were densely packed in the simulated area because that best represented a small unit in close combat and because of the difficulty in using large terrain databases.

The Simnet experience contrasts sharply with real exercises. Figure 1 shows the location of some of the 2,191 entities in an actual combat training scenario at the US Army National Training Center at Fort Irwin, California, replayed in the Janus Combat Model. Our analysis of this exercise showed that in the 10 hours of total maneuver, one third of the vehicles did not move. As the exercise progressed, over half of the vehicles became disabled and stopped all movement. Furthermore, 60 percent of the terrain was outside the detection range of all the vehicles.

## Exploiting reality

Increasing the number of entities by more than two orders of magnitude requires us to think beyond these artificial situations. We believe it is inappropriate to strictly extrapolate the Simnet and DIS experience (or any of the small-scale research virtual environments) to large VEs. Moreover, large VEs are likely to have domain-specific requirements such as strong data consistencies and process synchronization, which allow participants to manipulate objects together and interact closely. We can exploit aspects of the real world, such as areas of interest and movement rates, to efficiently use multicast groups, eliminate ESPDU keep-alive updates, enhance the reliability of large VEs, and reduce overall bandwidth requirements. We do this by grouping entities within spatial, temporal, and functional classes, and associating these classes with multicast networks.

### *Spatial classes*

In the real world, which virtual environments emulate, entities have a limited area of interest. For example, a tank on a battlefield can affect and observe other entities out to a range of less than 4 kilometers. On the other hand, a person on foot typically has an area of interest of only several hundred meters. This would be the case for a dismounted infantryman or a human simulated for a typical role-playing adventure game. Entities whose areas of interest overlap are members of a spatial class or group in the virtual environment.

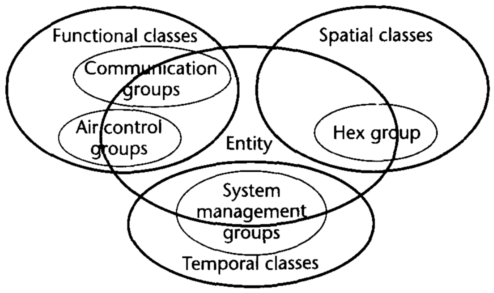
In a military simulation, membership in spatial classes would change relatively slowly. Helmbold studied advance rates and found that land combat operations stand still 90 to 99 percent of the time.[4] The world record for aggregate movement in modern warfare was 92 kilometers per day for four days (about 6 kilometers per hour) by the 24th Mechanized Infantry Division in Desert Storm.[5] Individual vehicles may move much faster, but they would not continue at high rates very long because they fight as part of units in which movement must be coordinated.

Our approach is computationally efficient—constant time versus $O(\log n)$ for simple collision detection using octrees or bounding volumes—and takes advantage of multicast networks for partitioning the environment.[6]

### *Functional classes*

Entities within a common functional class may communicate with other entities in that class. For example, simulated radio traffic should be restricted to only the interested parties of a functional group. Functional classes could also relate to system management or services, such as simulation host status messages.

For example, in a military simulation, a functional class might be an air control group, comprised of entities primarily concerned with other entities and events occurring in the air. Air defense and aircraft entities would make up most of this group. Aircraft and air defense systems are relatively sparse compared to other combat systems such as tanks. Each air defense system would also belong to a spatial class. Aircraft interested in a particular area of ground can focus and join the spatial group associated with its area of interest.



**2 Relationship between entity and multicast groups. One entity may belong to several spatial and functional classes, as well as a temporal class.**

### *Temporal classes*

Some entities do not require real-time updates of all state changes. A system management entity might only need updates every several minutes. Similarly, a simulator of a space-borne sensor only needs a general awareness of ground vehicle entities and therefore can accept low-resolution (that is, infrequent) updates. When it needs more resolution, the simulator, like aircraft entities, can focus and become part of a spatial group. Entities requiring updates at a similar rate may belong to similar temporal classes.

## Area-of-interest manager

We propose using a software "glue" between the DIS event and state PDU paradigm and the network layers, which are wedded to reality. The area-of-interest manager (AOIM) partitions the VE into a set of workable, small-scale environments or classes to reduce computational load on individual hosts, minimize communications on network tail links, and localize reliability problems. Furthermore, the AOIM is distributed at every simulator, to distribute partition processing among hosts.

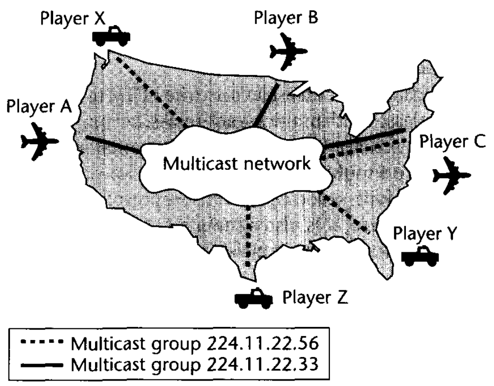
### *Multicast network groups*

The AOIM uses spatial, temporal, and functional classes to establish membership in multicast network groups, as shown in Figure 2. Multicast services allow arbitrarily sized groups to communicate on a network via one source transmission.

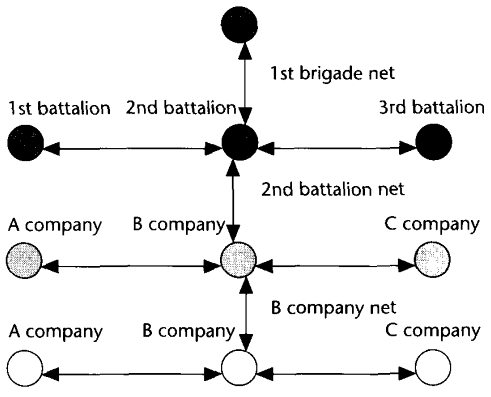
Multicast provides one-to-many and many-to-many delivery services for applications such as teleconferencing and distributed simulation, where we want to communicate with several other hosts simultaneously. For example, a multicast teleconference allows a host to send voice and video simultaneously to a set of (but not necessarily all possible) locations. With broadcast, data is sent to all hosts, while unicast or point-to-point routes communication between only two hosts.

The Internet Group Management Protocol provides an addressing scheme for an unreliable, connectionless, multicast service routable over the Internet.[6] For the AOIM, IP Multicast allows the creation of transient multicast groups that can be associated with an entity's area of interest (AOI).

In this context, IP Multicast addresses can essentially be used as context labels instead of physical destinations. Figure 3 shows this. Players X, Y, and Z send data to the IP Multicast group address 224.11.22.56 rather than explicitly forwarding packets to each and every player. The network takes over this requirement. Players A and B send and receive traffic relevant only to their group, 224.11.22.33, while C is a member of both and participates in each session.


Player X
Player B
Player A
Multicast network
Player C
Player Y
Player Z
····· Multicast group 224.11.22.56
—— Multicast group 224.11.22.33

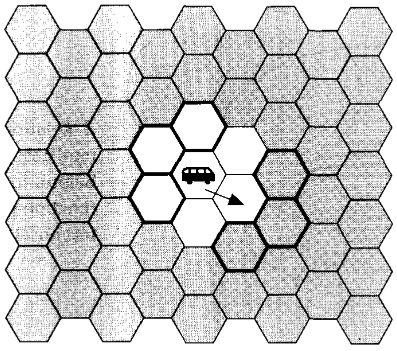
**3 Multicast communications in a simulation. Groups are expressed as IP Multicast addresses. Player C is a member of both multicast groups.**



**4 Radio communications in a functional class can be mapped to a channel group, as shown in this hierarchy of communication channels for a military organization.**

Therefore, multiplexing and demultiplexing is done at the network level. This separates traffic into classes such as audio, video, and simulation data. For example, the radio communications functional class can be mapped to a particular multicast group address or *channel* group (Figure 4). The network interface hardware filters appropriate multicast groups, so this does not consume processor cycles.

As stated before, such partitioning is necessary to reduce the enormous computational requirements of large (100,000 player) simulations. A 1990 Simnet exercise with 1,000 objects was limited not by network bandwidth (loads ran at 50 percent), but by local host processor performance.


5 Area of interest for a vehicle mapped to a subset of multicast groups.

### Associations

To illustrate our ideas, let's use the AOIM to associate spatial classes with multicast addresses. Let's say we partition the virtual environment into hexagonal cells, each mapped to a multicast group. Figure 5 shows a vehicle associated with seven cells, which represent its area of interest. Hence, it is also a member of seven network multicast groups. The vehicle's host listens to all seven groups but, with two exceptions (when joining or leaving groups), it sends PDUs only to the one associated with the cell it occupies.

We use hexagons for several reasons. First, they are regular, uniformly orientated, and uniformly adjacent. As the vehicle moves through the environment, it uniformly adds and deletes the same number of cells (and multicast groups). A vehicle's area of interest is typically defined by a radius—like a cellular telephone's transmitter signal. If we divided the VE into squares, we would either need to include more area than necessary (and thus include more entities in our area of interest) or use smaller grids—requiring more multicast groups—and compute which grids the vehicle should be associated with.

### Group changes

Entities can belong to several groups at once, to avoid boundary or temporal aliasing. A ground-based entity will likely have few group transitions within an hour because, on average, vehicle groups move slowly relative to the entire virtual environment. A vehicle moving at the Desert Storm record advance rate would traverse, on average, one cell each hour. When it moves cells, the vehicle in Figure 5 must join and leave three multicast groups associated with cells at the periphery of its area of interest, where change is less critical—ameliorating the effects of latency caused by joining and leaving new groups. The outlined clear cells are removed from its area of interest, and the outlined grey cells are added as the vehicle moves to a new cell.

We use group changes as an opportunity for database updates—similar to a paged-memory scheme—to eliminate regular ESPDU updates. We do this in a logical, distributed manner using knowledge about the age of entities with respect to their particular group.

An entity joins a group as a passive or active member. Active members send as well as receive PDUs within the group, are located in the cell associated with the group (that is, the center of seven cells), and can become the group leader. Passive members normally do not send PDUs to the group except when they join or leave. They are associated with the group because the cell lies within their area of interest, yet they do not occupy the central cell.

When an entity joins a new group, it notes the time it entered and issues a Join Request PDU to the cell group. The PDU has a flag indicating whether the cell is active or passive. The group leader replies with a Pointer PDU that references the request and in turn multicasts a PDU containing a pointer to itself or another active entity. The new member sends a Data Request PDU to the referenced source, which issues a Data PDU containing the aggregate set of active entity PDUs. A passive entity becomes an active member of a group by reissuing the Join Request PDU with a flag set to active when entering a cell. Departures from the group are announced with a Leave Request PDU.

The oldest member of the group is group leader, and we use timestamps to determine the oldest member. The first active member of a group will issue several Join Request PDUs before concluding that it is the sole member of the group and therefore the oldest. When a passive entity determines that there is no leader, it merely listens for active members. A new active member of an established group issues a Join Request PDU, receives the Data PDU, notes the join timestamps of the members, and keeps track of those who enter and leave.

### Rationale

The Data PDU may be sent reliably to the issuer of the Join Request PDU via a unicast protocol as a heavyweight object, that is, a large data set requiring reliable transfer. In a distributed simulation with many members, reliability, as provided in the Transmission Control Protocol, would normally penalize real-time performance merely by having to maintain timers for each host's acknowledgment.

Moreover, flow control is inappropriate for DIS, since systems with humans in the loop can recover from a lost state message more gracefully than from late arrivals. Fortunately, DIS tolerates some unreliability and mediates it with dead-reckoning and smoothing algorithms. Other applications, such as packet voice and video, use adaptive techniques to handle lost packets and delays.

However, we can reliably send the Data PDU because the entity will normally be joining a group at the periphery of its area of interest, where latency is not as critical. Learning about new members of entity groups under the DIS model typically takes at least five seconds while transiting through the virtual environment to a new active group. Assuming sufficient bandwidth, new-entrant learning can take less than one second under our architecture.

Furthermore, large VEs will naturally have some unreliability. Currently, an entire DIS simulation involving hundreds of entities can fail due to a single rogue application, because all communication is broadcast. One malfunctioning device or application can jam an entire simulation. Partitioning the VE into groups prevents problems from disrupting the entire simulation.

The area-of-interest manager can run as a separate thread or process and eliminates the need to change current DIS PDU semantics. The upper-level application simulating an entity need not know of the partitioning. Therefore, we can adapt many current DIS applications to support this architecture.

## Communications model

We conjecture that a large, real-time virtual environment cannot guarantee strong data consistency and reliable communication among all its participants simultaneously. Instead, four types of communication can be established which, used together, allow stronger consistency than simply broadcasting state messages. They provide a much richer world through a mechanism for sending large objects reliably and supporting VE partitioning.

**6 Medic conducting first-aid in NPSNet. The medic's movement is based on the University of Pennsylvania's Jack Motion Library, developed for the Army Research Laboratory.**

- *Lightweight interactions.* These messages are composed of the same state, event, and control PDUs used in the DIS paradigm but implemented with multicast. They are lightweight because the complete semantics of the message are encapsulated within the maximum transfer unit (MTU) of the underlying data link to permit asynchronous real-time interactive use. Therefore, these PDUs are not segmented. They are either received completely or not at all because they are communicated via connectionless and unreliable (unacknowledged data) networks. The MTU values are 1,500 bytes for Ethernet and 296 bytes for 9,600-bps Point-to-Point Protocol (PPP) links.
- *Network pointers.* We propose lightweight references to resources, similar to Uniform Resource Identifiers (URI) defined in the Hypertext Transfer Protocol (HTTP). Pointers are multicast to the group, and members cache them. Therefore, common queries need not be present, and the server can direct responses to other group members. We distinguish between pointers and lightweight interactions (for example, a Join Request PDU) because pointers do not completely contain an object, but rather its reference. Pointers provide a powerful mechanism for referencing not only the current aggregate state of the group but also terrain, model geometry, and entity behaviors defined by a scripting language. Network pointers in the World-Wide Web have revolutionized Internet communication.
- *Heavyweight objects.* These objects require reliable, connection-oriented communication. For example, an entity may require model geometry after joining a group not in its database. The entity multicasts a request for the geometry and receives in response a multicast pointer to the source. It then establishes a reliable network connection over which to receive the heavyweight object of the model geometry. As efforts such as the Virtual Reality Modeling Language (VRML) gain acceptance, heterogeneous systems may be able to exchange this type of information.
- *Real-time streams.* Video and audio traffic provide continuous streams of data that require real-time delivery, sequencing, and synchronization. Moreover, these steady streams will be long-lasting, persisting from several seconds to days. They are multicast on a particular channel to a functional class. In contrast to the current DIS protocol, we propose using pointers to direct entities to these channels, rather than forcing the virtual environment (which may be as simple as a text-based application) to receive both lightweight DIS PDUs and real-time video streams. Moreover, the VE can spawn a separate process that incorporates an adaptive receiver and thereby separates the handling of bursty simulation messages from real-time streams.

## Status of the work

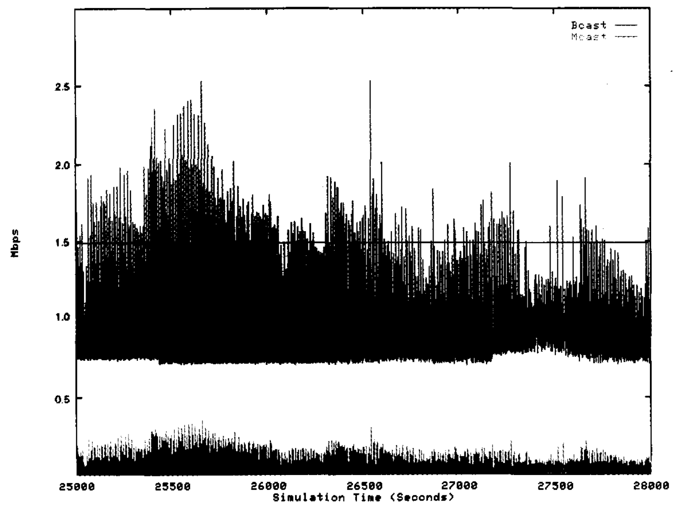
We have implemented an IP Multicast version of the NPSNet-IV 3D vehicle simulator using a network library developed by Paul Barham and John Locke that supports multiple threads and dynamic creation of multicast groups.[7] Furthermore, we are incorporating the algorithms to support the area-of-interest manager and have developed a simulation to predict and evaluate the results. NPSNet is widely used by universities, industry, and government for distributed virtual environment research. The Naval Research Laboratory is using NPSNet to explore large DIS. We are also collaborating with Sarcos Inc. and the University of Pennsylvania to insert humans into virtual environments. For example, Figure 6 shows an NPSNet simulation to develop medical training applications for a project sponsored by ARPA. NPS has developed a new DIS PDU to communicate the human articulations over the network.

The multicast version of NPSNet has been successfully tested over the Internet with several sites, including the Naval Research Laboratory, George Mason University, Massachusetts Institute of Technology, Sprint, Stanford Research Institute, and the Rand Corporation. We have also modified a version of the Modular Semi-Automated Forces (Modsaf) simulator developed by Loral to support multicast for generating many simulated military vehicles and aircraft.

## Simulation

We developed a DIS network simulation to examine these questions:

- Does partitioning using our architecture reduce bandwidth and computational requirements for large VEs as compared to the DIS model?
- Does the architecture scale, and if so how well?



**7 Mean peak bandwidth of multicast (4-kilometer hexagons) versus broadcast traffic with 2,191 entities.**

We were interested in the effect of entity distribution and maneuvering on our architecture—particularly as compared to the current DIS broadcast scheme. We collected entity behavior data from Janus, a large constructive model widely used by the US Army for research and training. The data was based on a real-world military scenario (similar to one proposed for STOW 97) and actual large-scale exercises at the National Training Center. After postprocessing, we applied our partitioning algorithms to the military entity.

Our simulation using spatial partitioning showed that, in a military context, as the number of entities increased, the mean peak bandwidth was less than T-1 rates, 1.5 Mbps. (See Figure 7.) The highest peaks represent the transfer of large data objects when entities transition among groups. This bandwidth will be quite feasible over networks to the home or office in the near future. AT&T and Intel, for example, plan to convert cable systems to provide 28-Mbps bandwidth to the home.[8]

The AOI architecture takes advantage of the fact that it does not use entity keep-alives or heartbeats for new entrants and reduces the bandwidth costs associated with them. Rather, new entrants are informed of the existence of other entities during the Join procedure. Furthermore, assuming that entities are distributed across different subnets, *multicast association reduces the traffic demands* on tail links by confining the scope of an entity's communication to its area of interest and implicitly directing its traffic to a subset of hosts on the network.

Perhaps more important than ameliorating bandwidth costs, partitioning can reduce the amount of state that an entity must maintain. This architecture, as opposed to the DIS model, scales with the increase in entities. For our simulation, using the National Training Center data with 4-kilometer-radius hexagons, the maximum entities in occupied areas of interest held relatively constant at approximately 1,800, as we increased the number of total entities from 2,191 to four times that amount (Figure 8). This number probably represents the worst case, consisting of a mixture of tanks and light infantry with their weapon systems and vehicles.

The peak number of entities using the architecture presents a feasible computational goal. Moreover, we can reduce the maximum AOI density by making our hexagons smaller or reduce the impact by doing application-level filtering.

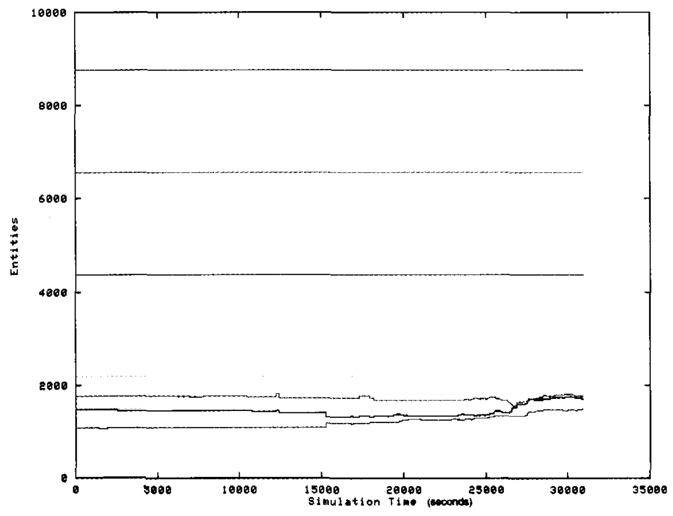
## Limitations and future research

Our work does not address all the problems of building large virtual environments for military simulations. First, this architecture may complicate developing secure environments because encryption devices may need to authenticate every other device for each multicast address.

Second, we have not analyzed the impact of fast-moving entities such as aircraft. We conjecture that this will not be a major obstacle for a number of reasons. Most aircraft fly too high or too fast to actually observe individual ground entities and establish an association with them except for air defense systems. In the case of a system like JSTARS (Joint Surveillance Target Attack Radar System), which tracks ground vehicles, we suggest that it can belong to the functional air group and could receive low-rate ESPDUs from the temporal all group.

For example, each ground entity might send an ESPDU to the all group every hour or every time it moved 5 kilometers. For 50,000 entities, this is roughly 13 PDUs per second. Low-flying aircraft like helicopters must normally hover or circle to acquire a target and fly at one-fifth the speed of fighters. Therefore, these aircraft can join the spatial groups associated with their target area. These actions and the effects of other types of entity behavior need exploration.

Third, we did not directly consider network topology in our simula-



**8 The maximum number of entities in an area of interest stayed constant at about 1,800, even as we increased the number of entities in the simulation from 2,191 to four times that amount.**

tions. We need to determine whether this architecture might be more appropriate for a network with many subnets having a single entity or host located at the site versus one with a handful of subnets having hundreds of entities represented on each host. Our data suggests the former, and in the future we need to use models such as those being developed by the Naval Research Laboratory to examine this issue.

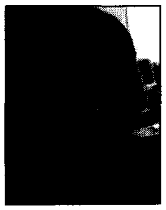
We also need to examine the impact of other partitioning methods, such as functional partitioning. In particular, we have not tested the impact of multimedia communication using this architecture, though we have used voice and video in conjunction with NPSNet. ■

## Acknowledgments

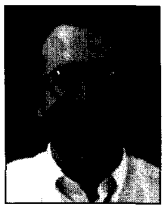
This work would not have been possible without the support of our research sponsors, the US Army Research Laboratories; Advanced Research Projects Agency; Defense Modeling and Simulation Office; US Army Simulation Training and Instrumentation Command; Headquarters, Department of the Army, Artificial Intelligence Center, Pentagon; and US Army Training and Doctrine Command Research Analysis Center.

## References

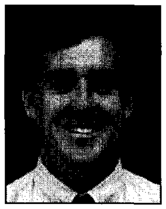
1. ANSI/IEEE Std 1278-1993—Standard for Information Technology, Protocols for Distributed Interactive Simulation, Version 1.0, IEEE, Piscataway, N.J., Mar. 1993.
2. R. Solitaire, "STOW-E To Do List," Dec. 1, 1994. Debriefing slides.
3. A.P. Pentland, "Computational Complexity Versus Simulated Environments," *Computer Graphics*, (Symp. Interactive 3D Graphics), Vol. 24, No. 2, Mar. 1990, pp. 185-192.
4. R.L. Helmbold, "Rates of Advance in Historical Land Combat Operations," Tech. Report CAA-RP-90-1, US Army Concepts Analysis Agency, Bethesda, Md., June 1993, pp. 5-2,3.
5. J. Dunnigan and R.M. Macedonia, *Getting It Right*, Morrow, New York, 1993, p. 211.
6. M.R. Macedonia and D.P. Brutzman, "MBone Provides Audio and Video Across the Internet," *Computer*, Vol. 27, No. 4, Apr. 1994, pp. 30-34.
7. M.R. Macedonia et al., "NPSNET: A Network Software Architecture for Large-Scale Virtual Environments," *Presence*, Vol. 3, No. 4, Winter 1994.
8. "On-Ramp for Info Highway Is Closer," *San Jose Mercury News*, May 5, 1995, p. D1.



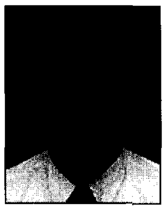
***Michael R. Macedonia*** *is a computer scientist whose research is directed toward the development of software architectures supporting large-scale distributed virtual environments. Macedonia received a BS from the US Military Academy in 1979, an MS in telecommunications from the University of Pittsburgh in 1989, and his PhD from the Naval Postgraduate School in 1995. He is a member of the IEEE Computer Society and IEEE Communications Society.*



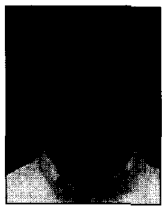
***Michael J. Zyda*** *is a professor of computer science at the Naval Postgraduate School and the Academic Associate and Associate Chair for Academic Affairs. His main research focus is in computer graphics, specifically the development of large-scale, networked 3D virtual environments and visual simulation systems. He is a member of the National Academy of Sciences' Committee on Virtual Reality Research and Development. He serves as the senior editor for* Presence. *Zyda received a BA in bioengineering from the University of California, San Diego, in 1976, an MS in computer science and neurocybernetics from the University of Massachusetts, Amherst, in 1978, and a DSc in computer science from Washington University, St. Louis, in 1984.*



***David R. Pratt*** *is an assistant professor of computer science at the Naval Postgraduate School. His PhD work dealt with the construction and management of complex, multiplayer environments. His interests include real-time graphics, terrain databases, and combat modeling.*



***Donald P. Brutzman*** *is a computer scientist in the Interdisciplinary Academic Group at the Naval Postgraduate School. His research interests include underwater robotics, real-time 3D computer graphics, artificial intelligence, and high-performance networking. A retired submarine officer, he received his BSEE from the US Naval Academy in 1978 and his PhD from the Naval Postgraduate School in 1994. He is a member of the IEEE, ACM Siggraph, AAAI, the Marine Technology Society (MTS), and the Internet Society (ISOC).*



***Paul T. Barham*** *is supervisory computer scientist for the NPSNet technical research staff at the Naval Postgraduate School. His research interests include true 3D imaging, spatialized audio, and large-scale, distributed virtual environments. He received his BS in computer science from North Carolina State University in 1987, and his MS in computer science with an emphasis on stereo computer graphics in 1991. He is a member of IEEE, ACM, and the Society for Information Display.*

*Contact the authors at the Naval Postgraduate School, Computer Science Department, Monterey, California 93943-5118; e-mail {macedonia, zyda, pratt, brutzman, barham}@cs.nps.navy.mil.*