Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments

*Michael R. Macedonia, *Michael J. Zyda, David R. Pratt, Donald P. Brutzman, Paul T. Barham*
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5118 USA
+1-408-656-2305
{macedonia, zyda, pratt, brutzman, barham}@cs.nps.navy.mil

ABSTRACT

We describe a network software architecture for solving the problem of scaling very large distributed simulations. The fundamental idea is to logically partition virtual environments by associating spatial, temporal, and functionally related entity classes with network multicast groups. We exploit the actual characteristics of the real-world large-scale environments that are 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). Finally, we present an example of how we would implement this concept for ground vehicles. We have begun design and construction of the AOIM for use with the NPSNET 3D vehicle simulator. NPSNET is currently the only Distributed Interactive Simulation (DIS) protocol compliant simulator using IP Multicast communications and is suitable for operation over the Internet.

KEYWORDS: Virtual Reality, Distributed Interactive Simulation, Internet Protocol Multicast, Distributed Interactive Entertainment, Large-scale Virtual Environments.

NETWORKED VIRTUAL WORLDS

The development of multi-user networked virtual worlds has become a major area of interest to the graphics community. The realization of 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) has 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.

Our group has been motivated to expand the capabilities of simulations and virtual environments (VEs) by exploiting multicast networks to serve medium to large numbers (more than 1,000) of simultaneous users. In particular, distributed interactive entertainment applications such as multiplayer games, whether in-home or location-based, will require scalable network architectures in order to provide both rich environments and profitable returns. This paper outlines the problems of scaling the most influential 3D distributed VE architectures, DIS and SIMNET, and our work in developing a solution to the design and construction of large-scale distributed simulations. In particular this paper addresses the networking software architecture for large-scale virtual environments (VEs).

Computer graphics applications are fundamentally concerned with visual bandwidth, i.e. the transfer of visual information to a human user. Visual bandwidth factors include image complexity, rendering speed, interaction latency and computational complexity. In this paper we examine how distributed applications can best exploit global network connectivity in order to drive the visual bandwidth of interactive 3D graphics for large-scale virtual environments.

PRACTICAL PROBLEMS WITH THE DIS PROTOCOL

Advances in computer architectures and graphics, as well as standards such as the IEEE 1278 Distributed Interactive Simulation (DIS) and BBN SIMNET protocols have made small scale (today, less than 300 interactive players) realistic man-in-the-loop simulations possible [1]. These standards have been used by the military for several years. Unfortunately, SIMNET, which was developed for small unit training, and its descendant, DIS, are currently not suitable for large-scale multiplayer VEs.

The following is a list of several major problems associated with scaling the current suite of DIS protocols in order to illustrate the difficulties of building large-scale VEs:

Enormous bandwidth and computational requirements for large-scale simulation. In schemes such as SIMNET and DIS, a simulation with 100,000 players could require on the order of hundreds of megabits per second of network bandwidth to each computer participating in the simulation, an unrealistic requirement for an affordable system in this decade. 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 U.S. Army's Synthetic Theater of War (STOW) exercises have shown this to be the case [10].

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 (i.e. collision detection and constraint satisfaction) will introduce a rapid rise in computational and spatial complexity with even modest size VEs [5].

We conjecture that on the order of 1000 entities are the upper bound limit to which a single host can realistically manage in real-time despite future advances in computer and graphics architectures because of 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 (e.g. 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 identically to bursty simulation traffic, i.e. through allocation of buffers and timing at the application layer.

Lack of an efficient method of handling static objects. Large numbers of static entities such as bridges and buildings may change with respect to an event (e.g. an explosion). These and other stationary objects must send update messages at regular intervals to inform the participants of their current state. For example, a tank that has been destroyed must constantly inform the world that it is dead to inform new entrants or other entities that may have missed the original state change message.

M**odels and world databases must be replicated at each simulator.** No mechanism in DIS exists to distribute objects on demand. For large-scale simulation this is a necessity, particularly when the simulators are heterogenous, controlled by different organizations, and 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 (e.g. a dismounted infantryman) on land normally does not need to concern itself with naval vessels, unless some unique scenario has the human near enough to the ocean so that it is visible.

REASONS FOR PROBLEMS

Event and State message paradigm. A basic requirement for DIS has been that the simulation of the VE must be, as a whole, stateless -- data is fully distributed among the participating hosts and entities are semi-persistent. Therefore, every entity must be made aware of every event (e.g. a missile detonation communicated by a Detonation Protocol Data Unit or DPDU) just on the chance it may need to know it. According to the protocol, an entity must, on a regular basis, communicate all of its state information (an Entity State Protocol Data Unit or ESPDU) to every member of the group - even though the data contained in the ESPDU is often redundant and unnecessary (e.g. aircraft markings). More importantly, these "keep alive" ESPDU messages can comprise 70% of the traffic for large-scale simulations.

This paradigm as applied in DIS does not take into consideration that different simulated systems have different real-world sensing capabilities that translate into each entity's VE data requirements. In a large VE, it is unlikely that two entities representing ground vehicles separated by 200 km need to be aware of each other. Yet, under the current architecture they must inform each other of state changes and updates.

The rationale for this is to avoid the reliability problems of a central server, to simplify communication protocols, and minimize latency while guaranteeing that hosts entering a simulation would eventually build their entity database through entity state and event messages. Furthermore, the use of broadcast ESPDU updates is part of the effort to maintain consistent view among the simulators within a particular tolerance.

Real-time system trade-off's. Reliability (guarantees that data sent is received) normally is compromised for real-time performance in large distributed groups. This is because in order to be truly reliable the system requires the use of acknowledgment schemes such as the one used in Transport Control Protocol (TCP) which defeats the notion of real-time, particularly if a player host must establish a virtual connection with every other entity host to ensure that each received data correctly. Therefore, large-scale environments must rely on connectionless (and therefore unreliable) network protocols such as the User Datagram Protocol (UDP) for wide-area communications.

The corollary is that a real-time environment should avoid transactions between individual entities since this requires reliable communications. Furthermore, schemes that use a central database do not work well in a large VE 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 the speed of modems (less than 28 Kbps).

No "middleware" layer. There does not exist a DIS protocol component that mediates between distributed VE applications and the network. The current DIS paradigm implies the use of a bridged network because every message is broadcast to every entity. However, internetworking (routing over the network layer) is necessary for large-scale simulations because it provides the capability to use commercial services as opposed to private networks to bring together diverse, geographically dispersed sites; use different local network topologies and technologies (e.g. Ethernet and FDDI); and take advantage of "rich" topologies for partitioning bandwidth, providing robustness and optimization of routes for minimizing latency. Confining DIS to the data link layer requires the use of bridges which are an order of magnitude slower to reconfigure after a topological change than routers while the number of stations are limited to the tens of thousands. A network with routers is limited to the numbers accommodated by the address space.

Origins as small unit training systems for Local Area Networks (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 insure that the VE components have homogenous sets of models and terrain databases by replicating them at each host. The lack of middleware stems from the monolithic nature of these small scale environments which could be distributed using a single LAN. Hence, *broadcast* communication was sufficient for these limited environments.

These origins have also influenced the current assumptions about the density and rates of activity of entities in large-scale simulations that do not necessarily match the real world. Players in SIMNET participated for short periods (several hours) and were highly active because the purpose of the simulation was to train crews in coordinated drills. Furthermore, the density of entities with respect to the simulated area of play was high because that best represented a small unit engaged in close combat and because of the difficulty in using large terrain data bases.

The SIMNET experience contrasts sharply with real large-scale exercises. Figure 1 depicts the location of some of the 2191 entities from an actual combat training scenario at the U.S. Army National Training Center (NTC), Ft. Irwin, CA and replayed in the Janus Combat Model. Our analysis from this exercise showed that in the ten 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% of the terrain was outside the detection range of all the vehicles.

Figure 1. An armored brigade in the attack at the National Training Center, Ft. Irwin, CA. The area is 60 x 50 km. Friendly vehicles are blue; enemy vehicles are red.

EXPLOITING REALITY

Increasing the number of entities by more than two orders of magnitude requires us to think beyond these artificial situations. We believe that it is incorrect to strictly extrapolate the SIMNET and DIS experience (or any of the small-scale research VEs) to large-scale VEs. Moreover, large VEs are likely to be domain specific in their requirements. 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-scale VEs, and reduce overall bandwidth requirements.

In the real world, which virtual environments emulate, entities have a limited area of interest. For example, a tank on a battlefield can effect and observe other entities out to a range of less than 4 km. 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. The entities whose areas of interest overlap are members of a *spatial class* or group in the VE.

With respect to the military domain, group membership within these classes would change relatively slowly. Helmbold in his study on the rates of advance rates for land operation found that land combat operations stand still 90-99% of the time [2]. The worlds record for aggregate movement in modern warfare was 92 km/day for 4 days (or about 6 km/hour) by the 24th Mechanized Infantry Division in Desert Storm [3]. 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.

RELATED WORK

The partitioning of virtual worlds into spaces is a common metaphor for VEs. Multi-User Dungeons (MUDs) have used this idea and projects like *Jupiter* from Xerox PARC have extended this to associating “rooms” with multicast video and audio teleconferences [7]. Benford has described a concept for the spatial interaction of objects in a large-scale VE [8]. The 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 the transitions between awareness levels should occur [9]. The MASSIVE 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 [8]. This may be changing as their developers pursue the use of multicast communications and weaker data consistency.

APPROACH

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 [4]. Additionally, we consider two other criteria for establishing relevance among entities and their communication in the VE.

Entities also may belong to a *functional class* in which an entity may communicate with a subset of entities. Therefore, simulated radio traffic should be restricted only to the interested parties of the group. Other types of functional classes could be related to system management or services such as time synchronization.

Another example of a functional class in the military domain would be a VE “air control” group. The group would include entities that are primarily concerned with entities or events occurring in the air. Therefore, air defense and aircraft entities would comprise the majority of the group. Aircraft and air defense systems are relatively sparse in the whole as compared to other combat systems such as tanks. Air defense systems would also belong to a small subset of the spatial class. Aircraft which are interested in a particular area of ground can “focus” and join a spatial group associated with its area of interest.

Finally, entities can belong to a *temporal class*. For example, 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 updates. When there is a need for more resolution, the simulator, like aircraft entities, can focus and become part of a spatial group.

DIS AREA OF INTEREST MANAGER

We propose the use of a software “glue” between the DIS event and state PDU paradigm and the network layers that is 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 hosts, minimize communications on network tail links, and localize reliability problems. Furthermore, the AOIM exists with every simulator to distribute partition processing among hosts.

MULTICAST

The AOIM uses spatial, temporal, and functional classes for establishing membership in multicast network groups as shown in Figure 2. Multicast services allow arbitrarily sized groups to communicate on a network via a single transmission by the source. Multicast provides one-to-many and many-to-many delivery services for applications such as teleconferencing and distributed simulation in which there is a need 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) locations. With broadcast, data is sent to all hosts while unicast or point-to-point routes communication only between two hosts.

The Internet Group Management Protocol (IGMP) provides an addressing scheme for an unreliable, connectionless, multicast service that is routable over the Internet [4]. From the perspective of 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.

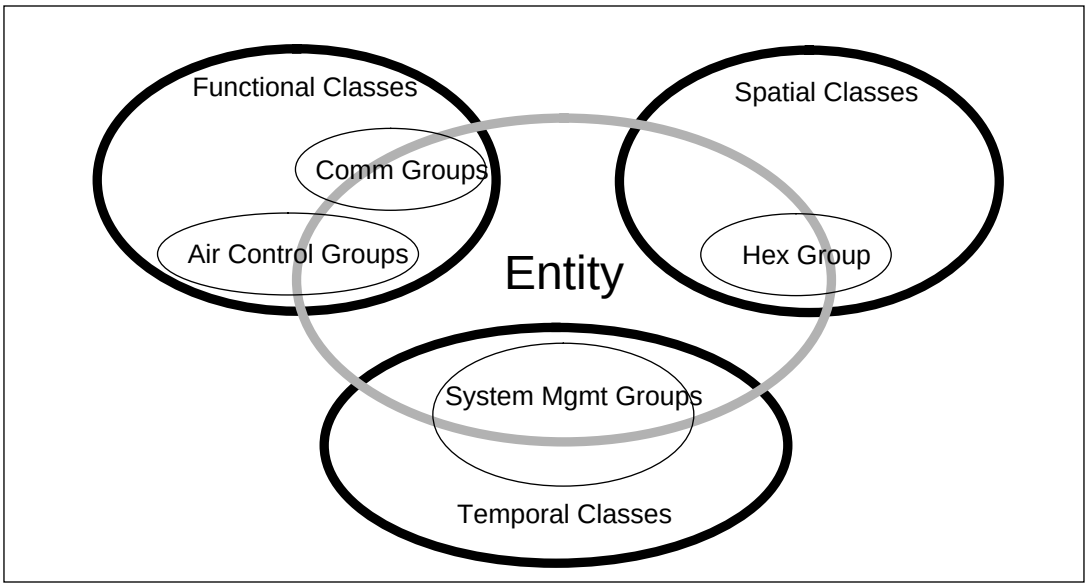

Figure 2. Relationship between entity and multicast groups.

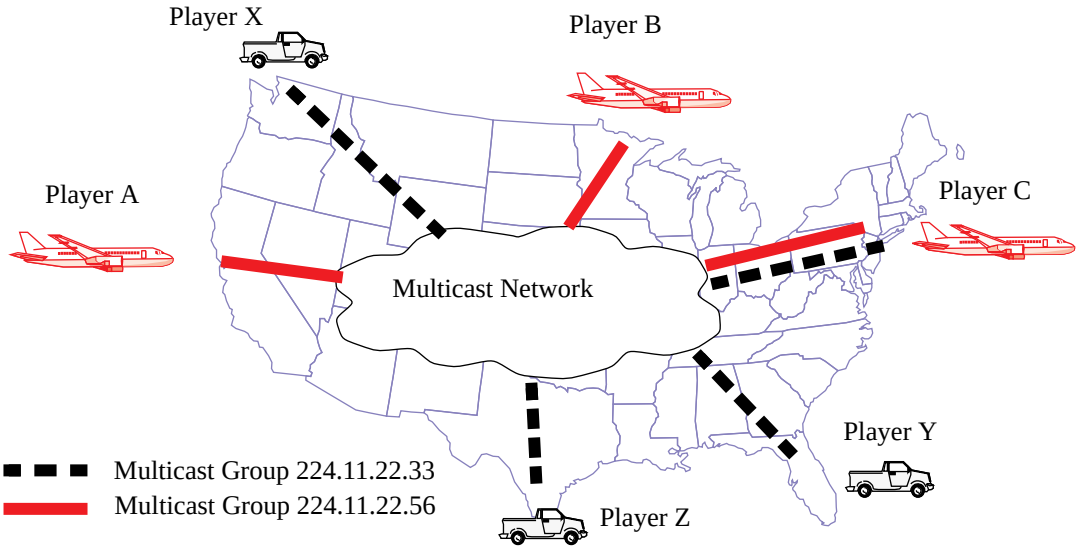

Figure 3. Simple illustration of multicast communications. Groups are expressed as IP Multicast Addresses. Note that Player C is a member of both multicast groups

Therefore, multiplexing and demultiplexing is done at the network level. This naturally provides a way of separating classes of traffic such as audio, video and simulation data. For example, the radio communications functional class would be mapped to a particular multicast group address or “channel” group (Figure 4). Filtering of appropriate multicast groups occurs in the network interface hardware and does not consume processor cycles.

As stated before, such partitioning is necessary to reduce the enormous computational requirements of large-scale (100,000 player) simulations. For a 1000 object exercise conducted in 1990 with SIMNET, the limiting factor was not network bandwidth, with loads running at 50%, but the local host processor performance.

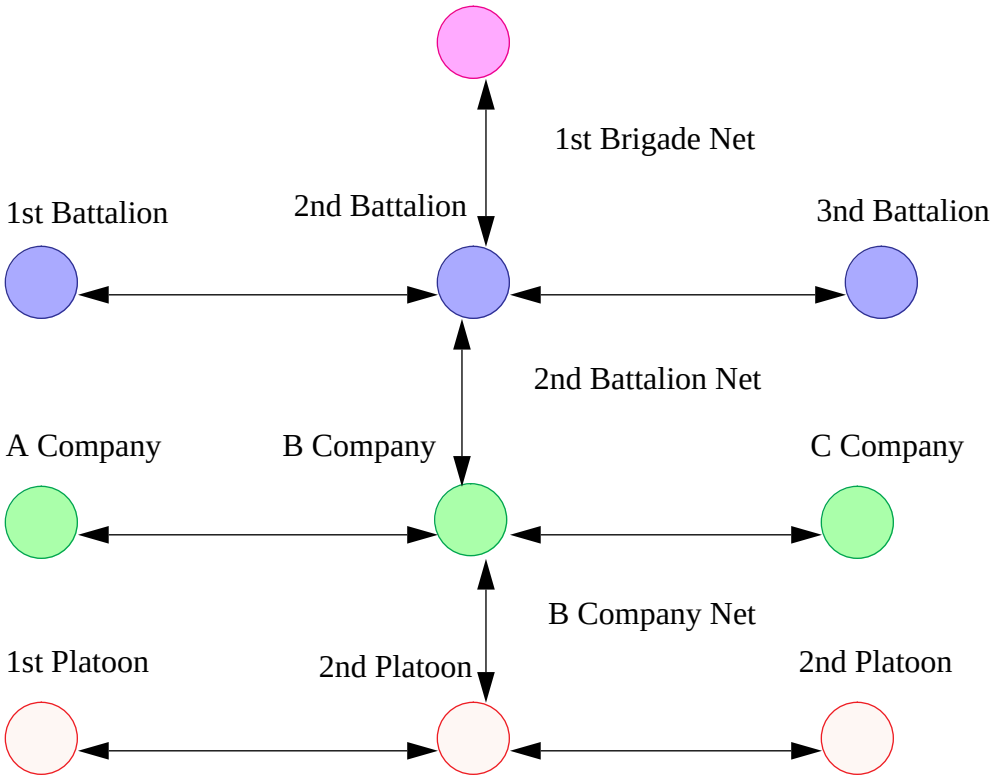

Figure 4. Military communication networks.

ASSOCIATIONS

To illustrate our ideas, we examine using the AOIM to associate spatial classes with multicast addresses. We suggest for this example partitioning the VE with appropriately sized hexagonal cells. Each cell is mapped to a multicast group. In Figure 5 we associate a vehicle with seven cells that represent its AOI. Hence, it is also a member of seven network multicast groups. The entity's host listens to all seven groups but, with two exceptions, it sends PDUs only to the one associated with the cell in which it is located.

There are several reasons hexagons are used. First, they are regular, have a uniform orientation, and have uniform adjacency. As the vehicle moves through the VE, it uniformly adds and deletes the same number of cells/multicast groups. A vehicle's AOI is typically defined by a radius - much like signal of transmitter in a cellular telephone system. If squares were used, we would either need to include more area than was necessary (and thus include more entities in our AOI) 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 a time to avoid boundary or temporal aliasing. There will likely be few group transitions by a ground-based entity within an hour because, on average, groups of vehicles will move slowly relative to the entire VE. If a vehicle was moving at the Desert Storm record advance rate, it would transition on average a cell once an hour. The vehicle portrayed in Figure 5 must join and leave three multicast groups which are associated with cells at the periphery of its AOI where change is less critical - ameliorating the effects of latency caused by joining and leaving new groups. The outlined clear cells are removed and the outlined grey cells are added as the entity transitions to a new cell.

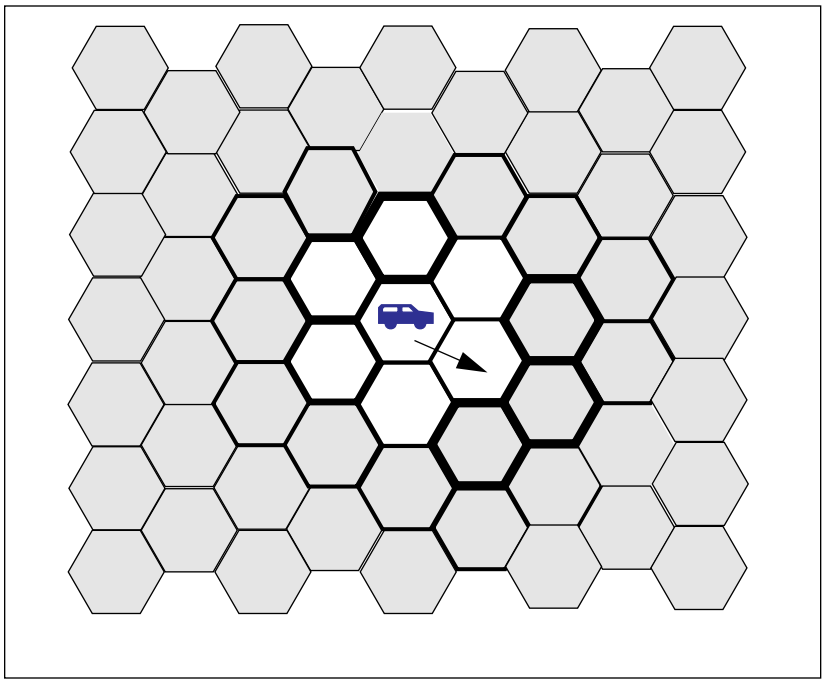

Figure 5. Area of Interest for vehicle mapped to a subset of multicast groups.

We use group changes as an opportunity for database updates -- similar to a paged memory scheme -- in order 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, 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 is within their AOI, yet they are not located within the 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 it 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.

We use the oldest member of the group as the election method for group leader. We make use of timestamps to determine the oldest member. The first active member of a group will issue several Join Request PDUs before concluding that its 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 heavy-weight object. With a large member distributed simulation, reliability, as provided in the Transmission Control Protocol (TCP), would normally penalize real-time performance merely by having to maintain timers for each host's acknowledgment. Moreover, flow control is also not appropriate for DIS since systems with humans in the loop can recover from a lost state message more gracefully than from late arrivals. Fortunately, within the context of DIS, a certain amount of unreliability is tolerable and is mediated through the use of the dead-reckoning and smoothing

algorithms. Other applications such as packet voice and video can 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 that is at the periphery of its AOI where latency is not as critical. Learning about new members of entity groups under the DIS model takes at least five seconds while transiting through the VE to a new active group. Assuming sufficient bandwidth, new entrant learning can take less than a one second under our architecture.

Furthermore, large-scale VEs will naturally have some degree of unreliability. Currently, an entire DIS simulation involving hundreds of entities can fail due to a single rogue application because all communication is broadcast. In the DIS exercises it has been possible for a malfunctioning device or application to "jam" the simulation. Partitioning the VE into groups prevents problems from impacting on the entire simulation.

The AOIM can be run as a separate thread or process and eliminates the need to change current DIS PDU semantics. The upper-level application simulating an entity is not required to have knowledge of the partitioning. Therefore, many current DIS applications can be adapted to support this architecture.

COMMUNICATIONS MODEL

We conjecture that a large-scale real-time VE 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 for a much richer world through a mechanism for sending large objects reliably and supporting VE partitioning.

In our model there exists four methods for communication within the context of VEs:

Light-weight interactions. These messages are composed of the same state, event, and control PDUs used in the DIS paradigm but implemented with multicast. They are light-weight 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 1500 bytes for Ethernet and 296 bytes for 9600 Point-to-Point (PPP) links.

Network pointers. Proposed are light-weight references to resources, in a way similar to Uniform Resource Identifier (URI) as defined in the Hypertext Transfer Protocol (HTTP). Pointers are multicast to the group so that they can be cached by members. Therefore, common queries need not be resent and the server can direct responses to other members of the group. We make a distinction between pointers and light-weight interactions (e.g. Join Request PDU) because they 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. In the context of the World-Wide Web (WWW), network pointers have revolutionized Internet communication.

Heavy-weight objects. These objects require reliable, connection-oriented communication. For example, an entity may require model geometry after joining a group that does not exist in its database. The entity would multicast a request for the geometry and the response would be a multicast pointer to the source. The originating entity then receives the heavy-weight object via a reliable network connection. If efforts such as the Virtual Reality Modeling Language (VRML) are successful, 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 with the current DIS protocol, we propose the use of pointers which direct entities to these channels rather than forcing the VE (which may be as simple as a text-based application) to receive both light-weight DIS PDUs as well as video streams. Moreover, the VE can spawn a separate process which incorporates an adaptive receiver and thereby separates the handling of bursty simulation message from real-time streams.

STATUS OF WORK

We have developed 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 [6]. Furthermore, we are incorporating the algorithms to support the AOIM 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 (Figure 6). NRL is using NPSNET to explore large-scale DIS. We are also collaborating with Sarcos Inc. and the University of Pennsylvania to insert humans into VEs to support medical emergency and soldier skill training.

Figure 6. Medic conducting first-aid in NPSNET

The multicast version of NPSNET has been successfully tested over the Internet with several sites including Naval Research Laboratory (NRL), George Mason University, MIT, 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 the generation of large numbers of simulated military vehicles and aircraft.

SIMULATION

The simulation developed was used to examine these questions:

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

We were interested in the effect the behavior of entity distribution and maneuver had on our architecture -- particularly as compared to the current DIS broadcast scheme.The simulation was developed by collecting 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 use in STOW 97) and actual large-scale exercises at the NTC. After post-processing, we then took the military entity data and applied our partitioning algorithms.

Our simulation using spatial partitioning shows that, in a military context, as the number of entities increase with our architecture, the mean peak bandwidth was less than T-1 rates (Figure 7). The largest peaks are primarily caused by the transfer of large data objects when entities transition among groups. This will be quite feasible over networks to the home or office in the near future. For example, AT&T and Intel are planning to convert cable systems to provide 28 Mbps bandwidth to the home [11].

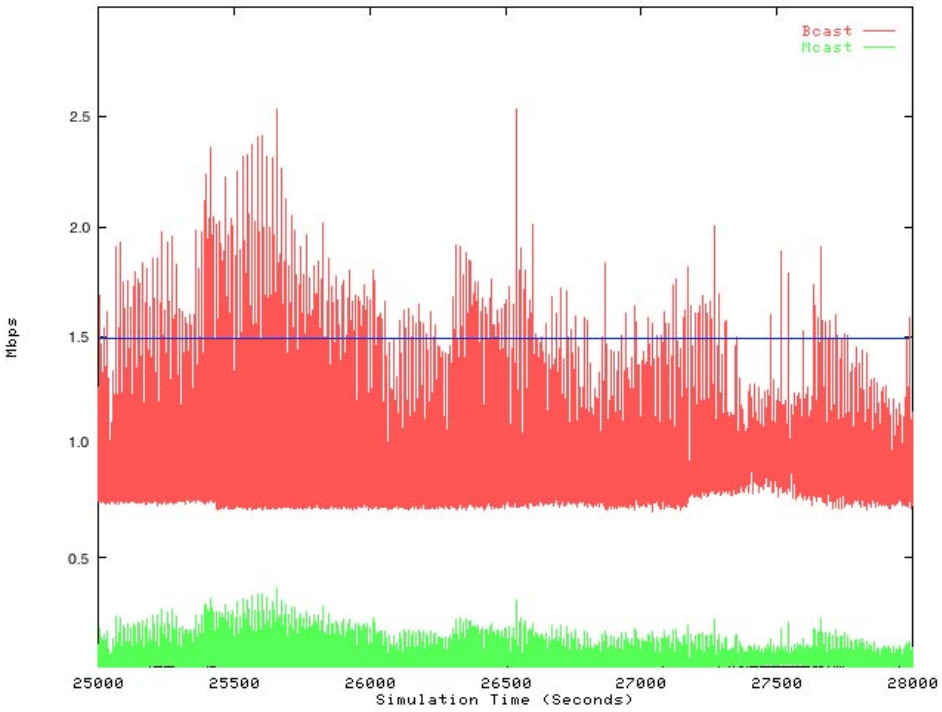

Figure 7. Mean multicast (4 km hex) vs. broadcast traffic with 2191 entities**.**

The AOI architecture takes advantage of the fact that it does not use entity keep-alives or heartbeats for new entrant learning to reduce 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. The architecture, as opposed to the DIS model, scales with the increase in entities. For our simulation, using the NTC data with four km radius's hexes, the *maximum* number of entities in an area of interest was relatively constant at approximately 1800 entities as we increased the number of entities from 2191 to four times that amount (Figure 8).This number probably represents the worst case with 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 hexes smaller or reduce the impact by doing application level filtering.

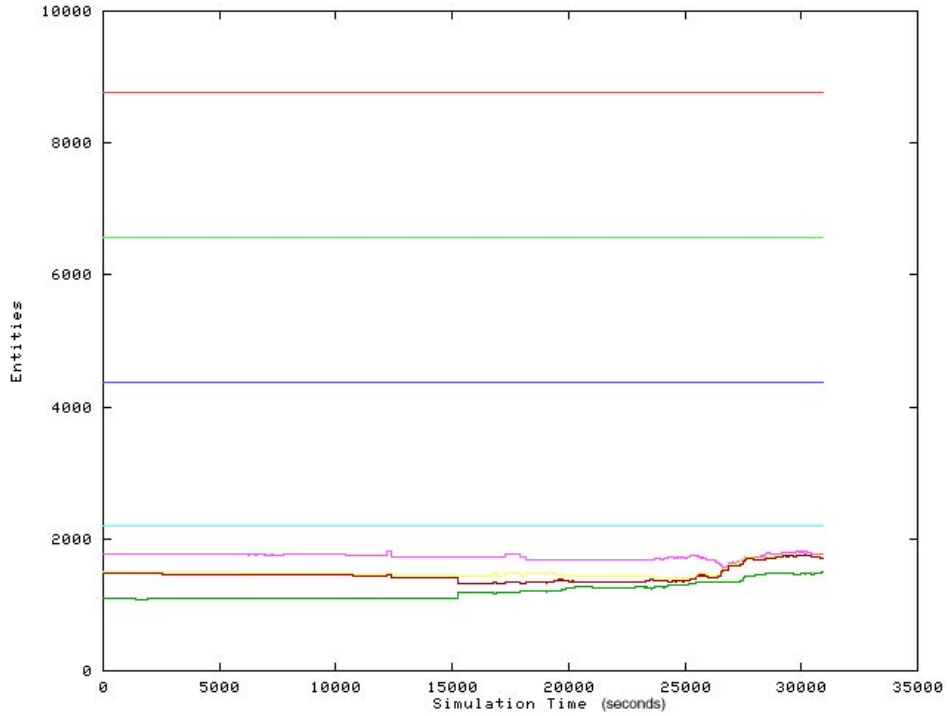

Figure 8. Comparison of max entities in AOI.

LIMITATIONS OF THE WORK

This work does not address all the problems of building large-scale VEs in a military environment. First, this architecture may complicate developing secure environments because encryption devices may need to authenticate every other device for each multicast address. Second, our work has 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 therefore establish an association with them except for air defense systems. In the case of a system like JSTARS which tracks ground vehicles, we suggest that it would belong to the functional “air” group and could receive low rate Entity State PDUs from the temporal “all” group.

For example, all ground entities could send an ESPDU every time it had moved five km or every hour to the all group. For fifty thousand entities this is roughly thirteen PDUs per second. Low flying aircraft like helicopters must normally hover or circle to acquire a target and fly at a 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 needs exploration.

Third, we did not directly consider network topology in our simulations. We need to determine whether this architecture may be more appropriate for a network with many subnets with a single entity or host located at the site versus one with a handful of subnets with 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 NRL 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.

CONCLUSION

This paper describes a concept that provides a network software architecture for solving the problem of scaling very large distributed simulations. The fundamental idea behind our approach is to logically partition virtual environments by associating spatial, temporal, and functionally related entity classes with network multicast groups. This is accomplished by exploiting the actual characteristics of the real-world large-scale environments that are to be simulated, and by focusing an entity's processing and network resources to its area of interest via an Area of Interest Manager.

ACKNOWLEDGMENTS

This work would not have been possible without the support of our research sponsors: USA ARL, ARPA, DMSO, USA STRICOM, USA HQDA AI Center-Pentagon, USA TRAC.

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

2. Helmbold, R. L. *Rates of Advance in Historical Land Combat Operations*, technical report, CAA-RP-90-1, US Army Concepts Analysis Agency, Bethesda, MD, (June 1993), pp. 5-2,3.

3. McQuie, R. *Historical Characteristics of Combat for Wargames*, technical report, CAA-RP-87-2, US Army Concepts Analysis Agency, Bethesda, MD, (July 1988), p. 13.

4. Macedonia, M.R., and Brutzman, D.P., MBone Provides Audio and Video Across the Internet. *IEEE Computer*, 27, 4 (April 1994), pp. 30-34.

5. Pentland, A.P., Computational Complexity versus Simulated Environments. *Computer Graphics. 1990 Symposium on Interactive 3-D Graphics* 24, 2 (March 1990), 185-192.

6. Macedonia, M. R., Zyda M.J., Pratt D.R., Barham P.T., Zeswitz S. NPSNET: A Network Software Architecture for Large-scale Virtual Environments. *Presence* 3,4 (Winter 94).

7. Curtis, P., Nichols, D.A. MUDs Grow Up: Social Virtual Reality in the Real World. 1994. *ftp://ftp.parc.xerox.com/pub/MOO/papers/MUDsGrowUp.ps*

8. Benford, S., Bowers, J., Fahlen, L. and Greenhalgh, C. Managing Mutual Awareness in Collaborative Virtual Environments. In *Proceedings of VRST '94*, World Scientific Publishing Company, NJ, pp. 223-236.

9. Carlsson, C. and Hagsand, O. (1993). DIVE - a Multi User Virtual Reality System. In *Proceedings of VRAIS '93* (September 18-22, Seattle, WA) IEEE, NJ, 1993. pp. 394-400.

10. Solitaire, Robert. STOW-E To Do List. (1 December 1994). Debriefing slides.

11. "On-ramp for info highway is closer", San Jose Mercury News, 5 May 1995, D1.