In Proceedings of VRAIS ‘95.

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.

INTRODUCTION

This paper outlines the problems and a proposed 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). We suggest a method that exploits the *spatial*, *functional*, and *temporal* relationships of real-world entities for partitioning VEs by associating network multicast groups with entity areas of interest.

The motivation for our effort is to expand the capability of virtual environments to serve large numbers (more than 1,000) of simultaneous users. Interest by the government, academic researchers, military, and telecommunications industry in large distributed virtual environments has been rapidly growing. 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.

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 (less than 300 players) realistic man-in-the-loop simulations possible [5,6,11]. 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.

PRACTICAL PROBLEMS WITH THE DIS PROTOCOL

We list 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 would require 375 Mbit per second (Mbps) of network bandwidth to each computer participating in the simulation, an unrealistic requirement for an affordable system in this decade[7]. 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 Simulated Theater of War (STOW) have shown this to be the case.

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 space complexity with

even modest size VEs [21].

We conjecture that 1000 entities are the limit to which a single host can realistically manage despite future advances in computer and graphics architectures.

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. allocation of buffers and timing at the application layer[20].

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 little coordination is expected prior to an exercise. Furthermore, it is not 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” messages can comprise 70% of the traffic for large-scale simulations [13].

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 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 on 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 [10].

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.

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 10 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 [16]. The world's 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 [15,17]. 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 [24]. Lockheed has developed a similar concept for spatial partitioning that assumes the use of ATM multicast "channels", i.e. mapping relevant groups to ATM's Virtual Channel Identifiers. Though ATM multicast technology is not yet mature, (few vendors support it), these ideas present exciting possibilities.

Benford has described a concept for the spatial interaction of objects in a large-scale VE [22]. 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 [26]. 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 [25]. This may be changing as their developers pursue the use of multicast communications and weaker data consistency.

APPROACH

Our approach is computationaly 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 [19]. 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.

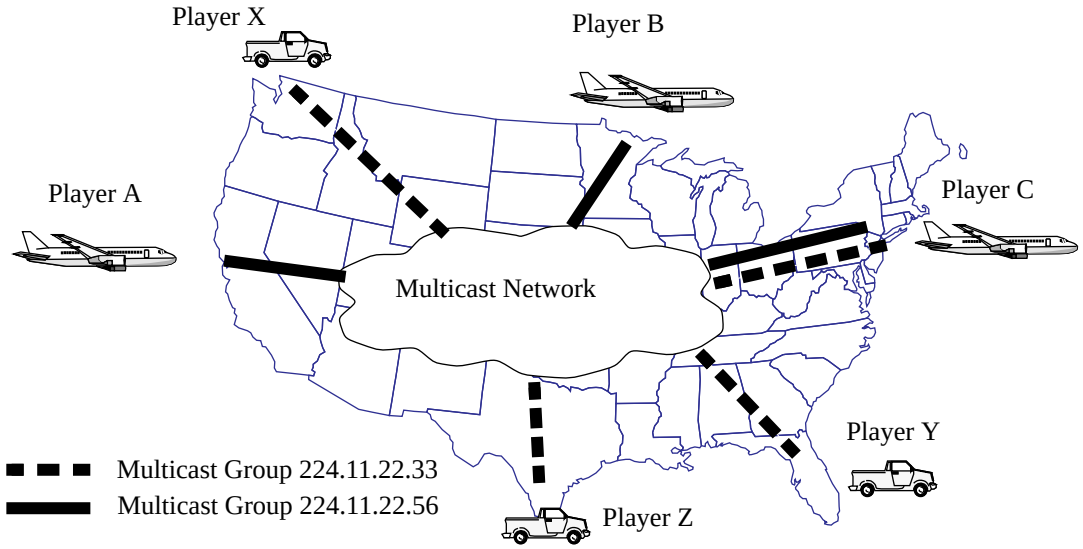

Figure 1. Simple illustration of multicast communications. Groups are expressed as IP Multicast Addresses. Note that Player C is a member of both multicast groups.

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 partitioning processing among hosts.

MULTICAST

The AOIM uses spatial, temporal, and functional classes for establishing membership in multicast network groups. Multicast services allow arbitrarily sized groups to communicate on a network via a single transmission by the source [10]. 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 Protocol (IP) Multicast protocol provides an addressing scheme that permits unreliable, connectionless, multicast service that is routable over the Internet [2,19]. 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 1 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,

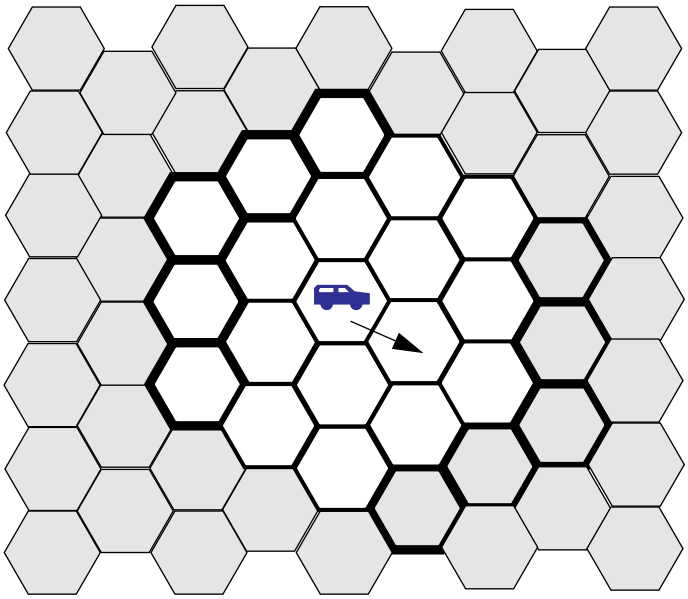

Figure 2. Area of Interest for vehicle mapped to a subset of multicast groups.

while C is a member of both and participates in each session.

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".

As stated before, this 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 [1]. Network simulations done by Van Hook have shown that a 90% reduction in traffic to a particular node is achievable for a 10,000 player exercise using multicast services [13].

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 associated with a multicast group. In Figure 2 we associate a vehicle with 19 hexagons that represent its AOI. Hence, it is also a member of 19 network multicast groups. The entity's host listens to all 19 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 we use hexagons. First, they are regular, have a uniform orientation, and have uniform adjacency [18]. As the vehicle moves through the VE, it uniformly adds and deletes the same number of cells/ multicast groups.

Secondly, 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. Using hexagons with a 2.5 km radius, the AOI above ranges from 12.5 to 8.6 km and the area is 411 km^2. If the average density of vehicles was 2 per km^2, then the entity host communicates with approximately 800 other entities. As mentioned previously, the AOI varies with respect to the capabilities of the system simulated.

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 2 must join and leave 5 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.

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 [4,8]. Other applications such as packet voice and video can use adaptive techniques to handle lost packets and delays [9]. 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.

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 in 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 for Ethernet is 1500 bytes and 296 bytes for 9600 point-to-point (PPP) links.

Network pointers. Proposed are light-weight references to resources, in a similar way to Uniform Resource Identifier (URI) as defined in the Hypertext Transfer Protocol (HTTP)[27]. 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 the 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 a 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, 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. If efforts such as the Virtual Reality Modeling Language (VRML) are successful, heterogeneous systems may be able to exchange this type of information [28].

Real-time streams. Video and audio traffic provide continuous streams of data that require real-time delivery, sequencing and synchronization. Moreover, these streams will be long-lasting, persisting from several seconds to days. They are multicasted on a particular "channel" to a functional class. In contrast with the current DIS protocol, we propose the use of pointers to direct entities to these channels rather than, for example, 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 which separates the handling of bursty simulation message from real-time streams.

ENTITY INTERACTIONS

Entities can only interact if they are aware of and can communicate with each other. Entity A becomes aware of entity B only if B is an active member of a group that A belongs to -- and therefore, in the AOI of A. If both are only passive members of the same groups then each one is beyond the view or influence of the other.

In a combat simulation, it is possible that if tank A fired a non-guided munition (which is not instantiated as an entity) at tank B, then B's AOI might not overlap the cell in which A was an active member tank. A must become an active member of the target area cell and forward a detonation PDU to that cell. According to the DIS protocol, entities assess for themselves the effects of the detonation and report via an ESPDU any state changes which are the result.

ADVANTAGES

Reduced latency for new entrant learning. Assuming an even distribution of entities in our example, for each cell joined an entity must receive data for about 40 other entities--approximately 40 Kbits. At 10 Mbps data transfer rates, it would take 4 ms to update a new entrant versus 5 seconds under the current DIS scheme.

Reduced bandwidth requirements. This architecture eliminates the need for entity keep-alives. New entrants are informed by the Join procedure of who exists in their particular groups. Multicast association further reduces the traffic demands on the tail links by confining the scope of an entity's communication to its area of interest and implicitly directing it traffic to a subset of hosts on the network.

No need for a centralized server. Using the oldest member of a group to serve Join requests is logical because it is the entity that should know all of the other entities and the past events that have occurred in the group. We expect that serving the group will be relatively undemanding with respect to Input/Output processing for the group leader because of the small number of active members in a group/cell and relatively slow transitions due to the expected real world transition rates for vehicles. Moreover, the server, through the pointer mechanism, can assign other entities to the task of serving the request. This provides an opportunity for exploring different algorithms for load balancing purposes.

Solves the static and dead entity problem. Likely candidates for the group leader will be static entities such as those representing buildings or bridges which can change state (i.e. collapse). Servers for these destructible entities will be the originating members of a spatially associated group and remain with the group for its entire existence. Moreover, static or dead entities are no longer a major burden to the VE with respect to wasting bandwidth with update ESPDUs. They need only to transmit PDUs upon initialization and when changing state.

Localization of reliability problems. large-scale VEs will naturally have some degree of unreliability. Partitioning the VE into groups prevents problems from impacting on the entire simulation. Currently, an entire DIS simulation involving hundreds of entities can fail because of a single rogue application because all communication is broadcast.

Maintains the current DIS semantics. The AOIM can be run as a separate thread or process and eliminates the need to change current DIS PDU semantics. The application simulating an entity is not required to have knowledge of the partitioning or the AOIM.

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 [23]. Furthermore, we are including the algorithms to support the AOIM concept presented here and developing a simulation to predict the results.

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.

Finally, we present an example of how we would implement this concept for spatial classes. We have begun design and construction of the AOIM for use with the NPSNET 3D vehicle simulator. NPSNET is currently the only DIS compliant simulator using IP Multicast communications and is suitable for operation over the Internet.

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.

RESOURCES

Many of the references noted below are available via the NPSNET Research Group's WWW home page:

ftp://taurus.cs.nps.navy.mil/pub/NPSNET_MOSAIC/npsnet_mosaic.html

1. Chung, J.W., *An Assessment and Forecast of Commercial Enabling Technologies for Advanced Distributed Simulation*, technical report, Institute for Defense Analysis, Arlington, VA (October 1992).

2. Deering, S. *Host Extensions for IP Multicasting*. RFC 1112 (Aug 1989).

3. Doris, K. Issues Related to Multicast Groups. In *Proceedings of the Eighth Workshop on Standards for the Interoperability of Defense Simulation* (March 1993), pp. 269-302.

4. Harvey, E.P., Schaffer, R.L., *The Capability of the Distributed Interactive Simulation Networking Standard to Support High Fidelity Aircraft Simulation*, technical report, BMH Associates, Inc. and BBN Systems and Technologies, Norfolk VA, Cambridge, MA. (July 1992).

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

6. Institute for Simulation and Training, IST-TR-93-20, *Communication Architecture for Distributed Interactive Simulation (CADIS) [Final Draft]*, University of Central Florida, Orlando, FL. (June 1993).

7. Loral Systems Company, *Strawman Distributed Interactive Simulation Architecture Description Document Volume 1*, technical report, Advanced Distributed Simulation Technology Program Office, Orlando, FL (March 1992).

8. Miller, D.C., Pope, A.C., and Waters, R.M. Long-Haul Networking of Simulators. In *Proceedings of Tenth Interservice/Industry Training Systems Conference* (December 1989), p. 2.

9. Partridge, C. *Gigabit Networking*. Addison-Wesley. Reading, MA. 1994, pp. 191-193

10. Perlman, R. *Interconnections: Bridges and Routers*. Addison-Wesley, NY, 1992, p. 258.

11. Pope, A., BBN Report No. 7102, *The SIMNET Network and Protocols*, technical report, BBN Systems and Technologies, Cambridge, MA, (July 1989).

12. Pratt, D.R., *A Software Architecture for the Construction and Management of Real Time Virtual Environments*, dissertation, Naval Postgraduate School, Monterey, CA, (June 1993).

13. Van Hook, D.J. *Simulation Tool for Developing and Evaluating Networks and Algorithms in Support of STOW 94*. Presented for Scalability Peer Review, (August 1993).

14. Zyda, M.J., Pratt, D.R., Falby, J.S. Barham, P.T., Kelleher, K.M. The Software Required for the Computer Generation of Virtual Environments**.** *Presence*, **2**, 2 (Summer 1993) 130-140.

15. Dunnigan, J. and Macedonia, R.M. *Getting It Right*, Morrow. New York, NY, 1993, p. 211.

16. 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.

17. 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.

18. Samet, H. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley. Reading, MA, 1989, pp. 20-21.

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

20. Feldmeier, D. C., Multiplexing Issues in Communication System Design. In *Proceedings of ACM SIGCOMM '90* (September 1990), ACM, pp. 209-19.

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

22. Benford, S., Fahlen, L.E. and Bowers, John. Supporting Social Communication Skills in Multi-Actor Artificial Realities. In *Proceedings of The Fourth International Conference on Artificial Reality and Tele-Existence (July 14-15*, Tokyo, Japan), 1994, pp. 205-223.

23. 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).

24. 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.

25. 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.

26. 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.

27. Berners-Lee, T. Hypertext Transfer Protocol (HTTP), A Stateless Search, Retrieve and Manipulation Protocol, Internet Engineering Task Force Draft, ftp://nic.ddn.mil/internet-drafts/draft-fielding-http-spec-01.ps, (19 December 1993).

28. Pesce, M. The Virtual Reality Modeling Language. http://www.eit.com/vrml/vrmlspec.html.