A Taxonomy for Networked Virtual Environments

Michael R. Macedonia
Fraunhofer Center for Research in Computer Graphics

Michael J. Zyda
Naval Postgraduate School

We discuss virtual environments in the context of how to distribute network communications, views, data, and processes while emphasizing those aspects critical to scaling environments. Systems that demand strong data consistency, causality, and reliable communications while supporting real-time interaction are not likely to scale very well. Furthermore, geographically dispersed systems require high-speed, multicast communication.

In this article we discuss what to consider when building large-scale virtual environments. This frame of reference provides a common understanding of distributed VE systems' many components. Currently, there are relatively few examples of academic VE systems in which to apply the framework. Several practical factors have limited research into large-scale distributed virtual worlds. For example, immature network technology has relegated most distributed VEs to Ethernet local area networks (LANs). Large-scale VEs will need to use wide area networks (WANs) to expand both their geographical scope and number of participating hosts. Furthermore, until recently, real-time graphics performance was confined to specialized and very expensive computer image generators. Software development and graphics databases have also progressed slowly, and the interfaces for immersing the human into the environment have been primitive at best.

These problems are being overcome to a limited degree by high-speed inter-networks; low-cost, off-the-shelf graphics workstations; and standard graphics tools and libraries. Moreover, distributed VEs hold promise for new educational, training, entertainment, and medical applications. For instance, more than 20 companies, including Microsoft, are developing 3D computer gaming networks. However, many challenges remain that would appear trivial if these applications were not distributed across a variety of networks.

Taxonomy

Virtual environments mimic many aspects of operating systems. For example, the Human Interface Technology Laboratory's now-defunct system, Virtual Environment Operating Shell (VEOS), provided much of a distributed operating system's functionality with programming language services. However, we are primarily concerned with a single application and not a general-purpose computing environment. Therefore, the most important questions about VE software architectures we address here are

- What is distributed?
- What are the modalities of the distribution?
- Why is it distributed?

Network communication

Several network communication aspects are largely responsible for answering the three questions above. Figure 1 shows the primary dimensions for VEs—bandwidth, latency, distribution schemes, and reliability.

Bandwidth. We pay particular attention to the effect of bandwidth in this article because the available network bandwidth determines a VE's size and richness. As the number of participants increases, so do the bandwidth requirements. On local area networks, this has not proved a major issue because technologies such as standard Ethernet (10 Mbps) are relatively inexpensive and the number of users for LAN-based VEs limited. In contrast, for wide area networks, bandwidths have generally been limited to T1 (1.5 Mbps), but the potential user base is much larger through the Internet.

However, networks are now becoming fast enough to be true extensions to the computer's backplane and to develop distributed VR applications. Distributed VR can require enormous bandwidth to support multiple users, video, audio, and the exchange of 3D graphics primitives and models in real time. Moreover, the data mix requires new protocols and techniques to handle data appropriately over a network link. The technologies providing these gains in performance blur the traditional distinction between local area and wide area networks.

1070-986X/97/$10.00 © 1997 IEEE

There is also a convergence between networks that traditionally carried only voice and video over point-to-point links (circuit-switching) and those that handle packet-switched data. The actual number of VEs to take advantage of these high-speed networks has been small and associated with Grand Challenge (high-performance computing) problems. The Multidimensional Applications and Gigabit Internetwork Consortium (Magic) network is a gigabit-per-second ATM network that connects Minneapolis, Minnesota; Sioux Falls, South Dakota; Lawrence, Kansas; and Ft. Leavenworth, Kansas. Magic (http://www.ai.sri.com:80/magic) allows a military commander to see 3D photorealistic computer-generated images of a very large interest area in real time, both from ground level and from the air, using data stored in a remote database. These images can be generated from elevation data (digital elevation maps), aerial photographs, building models, and vehicle models whose positions are updated in real time via the Global Positioning System. For example, a terrain database of Germany viewed on a workstation in Kansas receives images from California texture-mapped onto the terrain in real time. The network provides trunk speeds of 2.4 Gbps and access speeds of 622 Mbps, allowing an application to use a supercomputer (CM-5) to process data from a database at a second location and display the results on a workstation at a third location. The NASA Computational Aerosciences Project plans to use high-speed networks to support visualization of large computational fluid dynamics data sets by distributing processing onto several supercomputers across the United States. Gigabit networks will move supercomputer-generated actual geometries to remote graphics workstations for rendering. Similarly, the Electronic Visualization Laboratory at the University of Illinois has used a combination of Ethernet, Fiber Distributed Data Interface (FDDI), and High-Performance Parallel Interface (HiPPI) networks to develop a distributed VE application. The operator navigates through the VE using a CAVE (Cave Automatic Virtual Environment, a system that projects images on three walls or a hemicube for simulating "walkthroughs"), connected to Silicon Graphics workstations for rendering and control, which in turn connect to a CM-5 for actual simulation.[1]

- Distribution
  - Broadcast, multicast, or unicast
- Latency
  - Lag, jitter
- Reliability
  - Acknowledgements, negative acknowledgements
- Bandwidth

Figure 1. Network communication issues for VEs.

Distribution. Some distribution schemes scale better than others. Figure 2 shows three such methods. 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 needing to communicate with several other hosts simultaneously. For example, a multicast teleconference lets a host 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 establishes communication between two hosts.

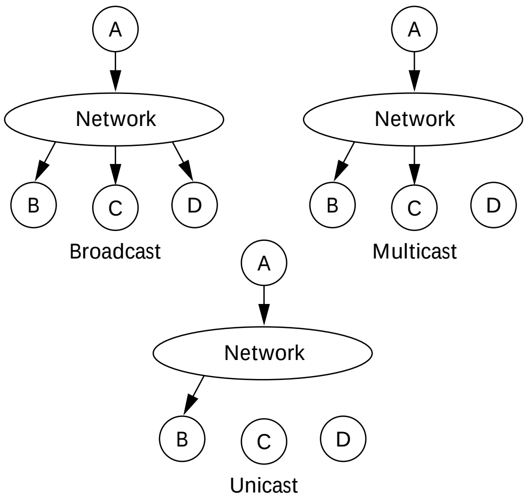


Figure 2. Examples of communication distribution methods—broadcast, multicast, and unicast.

Most distributed VEs employ some broadcast form—hardware-based, the Internet Protocol (IP), or point-to-point communications. For example, the MR Toolkit Peer Package, which creates distributed virtual reality applications over the Internet, originally used unicast for communications among the applications.[2] Unicast is also the general approach for Grand Challenge applications like Magic. The NASA project discussed above is another example where the network logically serves as a part of the visualization system, in a manner analogous to traditional image generators.

However, these schemes prove bandwidth inefficient for large groups. Furthermore, broadcast, which is used in SimNet—an early military Simulator Network developed by Bolt, Beranek, Newman (BBN) and Delta Graphics—and in most military Distributed Interactive Simulation (DIS) implementations, does not suit inter-networks because the network becomes flooded with unwanted traffic and cannot easily avoid routing loops. Moreover, IP broadcast requires that all hosts examine a packet even if the information is not intended for that host. This incurs a major performance penalty for that host because it must interrupt operations to perform this task at the operating system level. (SimNet uses the Ethernet's

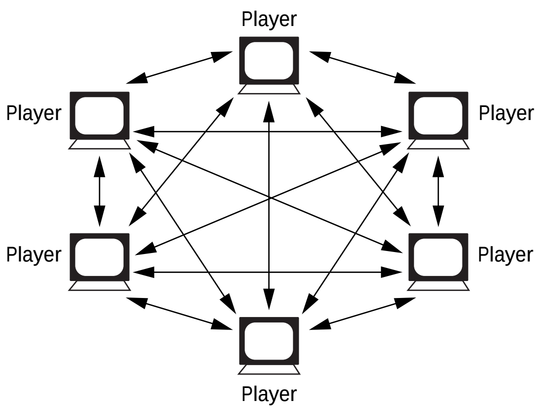


Figure 3. Point-to-point or mesh distributed model.

Figure 4. Spline-based MERL Diamond Park virtual environment.

hardware multicast capability, but only to create a single multicast group for the entire distributed simulation.) Point-to-point requires establishing a connection or path from each node to every other node in the network for a total of $n(n - 1)/2$ virtual connections in a group (see Figure 3). For example, with a 1,000-member group, each of the 1,000 individual hosts would have to separately address and send 999 identical packets.

Some researchers have proposed different ideas for using multicast to support VEs. Partitioning virtual worlds into spaces is a common metaphor for VEs. MERL, a Mitsubishi Electric Research Lab, developed the innovative Spline VE architecture. Spline uses multicast peer-to-peer communication. It also incorporates a region-based filtering scheme. These regions or "locales" partition the VE, while "beacons" provide entity updates about other objects in the VE.[3] MERL has implemented a prototype VE called Diamond Park using the Spline architecture (Figure 4).

Multiuser Dungeons (MUDs) have used the idea of spatialization. The Jupiter from Xerox PARC extended the concept to associating "rooms" with multicast video and audio teleconferences (ftp://ftp.parc.xerox.com/pub/MOO/papers/MUDsGrowUp.ps). Also at Xerox, Schilit and Theimer[4] developed an active map service that publishes object locations in a region using dynamic multicast groups associated with different region parts. For example, the system can track persons in a building by using active badges. Using multicast for updates reduces aggregate message traffic.

Benford described a concept for the spatial interaction of objects in a large-scale VE.[5] The spatial model uses different awareness levels between objects based on their relative distance and mediated through a negotiation mechanism. The Swedish Institute of Computer Science's Distributed Interactive Virtual Environment (DIVE)—an advanced experimental VE—implements this concept using "standard VR collision detection" to determine when transitions between awareness levels should occur.[6]

Others have suggested using multicast for DIS, but very few have actually conducted extensive research or implemented VEs using multicast communications. The Stanford Research Institute recommended multicast in an early 1990 white paper and it has been recommended for the IEEE 1278 standards group.

Van Hook at the MIT Lincoln Laboratories proposed using a grid-filtering combination to reduce the computational requirement of object filtering, an $O(n^2)$ operation.[7] Van Hook also suggested on-demand forwarding, in which entities would send a low-rate broadcast with terse state information. Each receiver would compute a range check and send state data to the visible entities. However, object-filtering and on-demand forwarding essentially establish a multicast group for every receiver. For example, in Figure 5, Entity 1 and 2 join each other's multicast groups. Entity 3 lies outside the range of 1 and 2 and therefore is a member of only its own group.

Until 1994, the DIS community resisted using IP Multicast because of the Defense Advanced Research Projects Agency's (Darpa) support for other network technologies, the lack of software architectures and algorithms that could exploit it, and limited hardware support. IP Multicast's status in the DIS world has changed. It is now part of the standard. Moreover, Pullen and others have suggested a two-level architecture using IP Multicast mapping to ATM multicast facilities.[8]

Latency. Another communication dimension, latency, controls the VE's interactive and dynamic nature—how well the players mesh in behavior. For a distributed environment to emulate the real world, it must operate in real time in terms of human perception. A key challenge is that the appropriate systems involving human operators must deliver packets with minimal latency and generate textured 3D graphics at 30 to 60 Hz to guarantee the illusion of reality. On top of this, introducing player communication services requires real-time audio, video, and imagery.

Latency poses a problem for network cue correlation. Both the delay of an individual cue (such as seeing an object move) and the variation in the length of the delay are important, particularly in closely coupled tasks that require a high degree of correlation (for example, flying in formation). This becomes a major challenge in systems that use wide area networks because of delays induced by long paths, switches, and routers. Network latency can be reduced to a certain extent by using dedicated links (or virtual ones using protocols like the Reservation Protocol), improvements in router and switching technologies, faster interfaces, and faster computers.

However, more bandwidth is not necessarily a complete solution. Operating at gigabit speeds presents a new set of problems. New methods for handling congestion are required because of the high ratio of propagation time to cell transmission time. By the time a computer in New York sends a message telling a host in San Francisco to stop sending data, it is too late to have stopped a gigabit worth of information from being transmitted.

The bottlenecks will most likely occur in the network interfaces, memory architectures, and operating systems of the computers on either end. The slow progress in increasing the FDDI's interface performance exemplifies the lag in technologies we will probably see as high-speed networks are fully deployed. Nor have memory speeds kept up with the leaps made in CPU and network performance. At the operating system level, most VR applications build upon commercial Unix versions not designed for real-time performance.

Other methods ameliorate latency effects. BBN developed dead-reckoning techniques for SimNet, which reduces a network's communications loads and perceived delays because of predictive modeling by the local host.[9] Briefly, a local entity passes state vectors to remote simulations. Both the local and remote simulations model the entity's probable path. The local entity sends update state vectors when its current location or orientation exceeds some predetermined error threshold from the modeled path. Singhal proposed a new dead-reckoning method that exploits position history.[10]

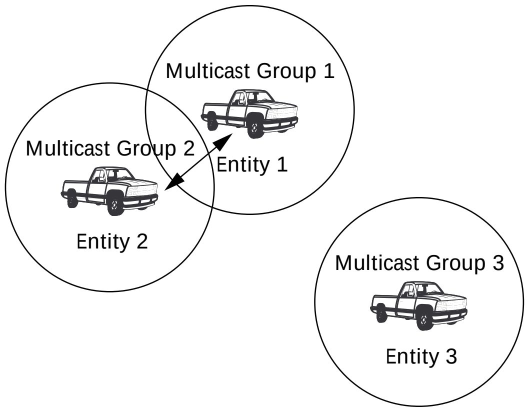


Figure 5. Multicast groups resulting from object-based filtering.

However, lag time can never be totally eliminated for widely distributed VE environments (for example, Earth to Mars). Therefore, we must use techniques such as synthetic fixtures, providing force and visual clues to operators in limited domains about that environment.[11]

Reliability. Finally, communications reliability often forces a compromise between bandwidth and latency. Reliability means that systems can logically assume that data sent is always received correctly, thus obviating the need to periodically resend the information. Unfortunately, to guarantee delivery, the underlying network architecture must use acknowledgment and error recovery schemes, which can introduce large amounts of delay—common in WANs and large distributed systems. Additionally, some transport protocols such as the Transport Control Protocol (TCP) use congestion control mechanisms unsuitable for real-time traffic—they throttle back the packet rate upon detecting congestion.

Real-time, reliable multicast protocols are currently not practical for large groups because guaranteeing that a packet is properly received by every host in the group requires an acknowledgment and retransmission scheme. With a large distributed simulation, reliability, as provided in TCP, would penalize real-time performance merely by maintaining timers for each host's acknowledgment and by holding up flow when a packet is lost for retransmission. Flow control introduces delays to the network to reduce congestion. Therefore, it is also not appropriate for DIS, which can recover from a lost packet more gracefully than from late arrivals—it is impossible for real-time simulations

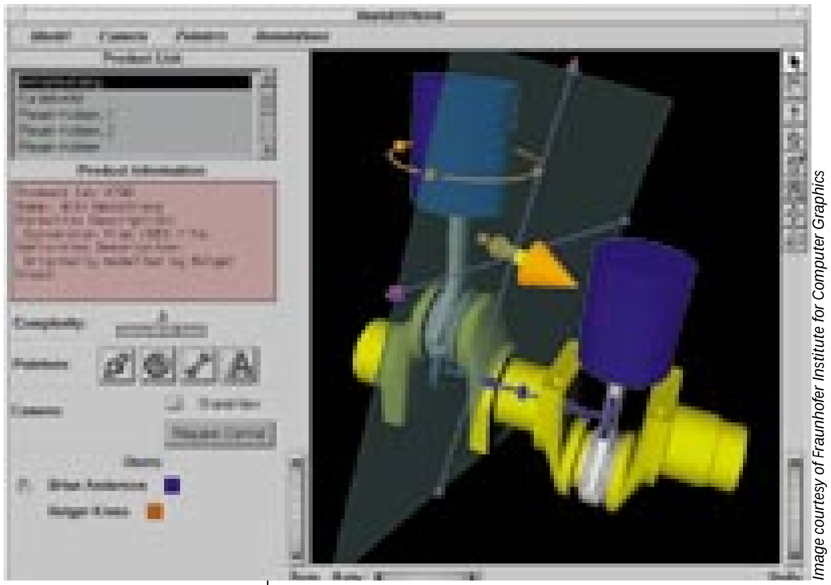


Image courtesy of Fraunhofer Institute for Computer Graphics

Figure 6. A shared 3D CAD viewer from Coconut.

to go backward in time. For example, when a packet is lost, the receiving host notifies the sender, possibly—due to propagation and network processing delay—invalidating a number of packets already sent. The sender must retrieve a copy of the lost packet and retransmit it. This also affects the windowing behavior, which in turn slows throughput.

However, researchers are trying to develop a reliable and scalable multicast service. The ISIS system, developed by Ken Birman, uses a reliable multicast service to guarantee that VE databases are accurately and synchronously replicated.[6] (A recent version of ISIS implements a reliable transport layer on top of IP Multicast.)

Whetten and Kaplan developed the Reliable Multicast Protocol (RMP), based on a token ring protocol that sits atop IP Multicast (http://www.mlds.com, http://research.ivv.nasa.gov/projects/RMP/). This method uses sequencing and negative acknowledgments (Nacks). The problem with this method is the potential for Nack implosions over the Internet, in which a group of receivers simultaneously send Nacks, adding to congestion and consequently causing the loss of more packets, introducing more Nacks. Again, reliable systems are not likely to operate in real time.

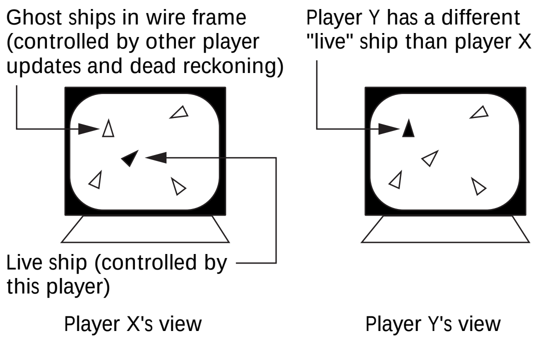

Ghost ships in wire frame (controlled by other player updates and dead reckoning)
Player Y has a different "live" ship than player X
Live ship (controlled by this player)
Player X's view
Player Y's view

Figure 7. Two views of a simulation from the local and remote perspective.

Netrek, an Internet multiplayer game that uses X Window system graphics, uses different degrees of reliability to gain better real-time performance. (Mark Pullen of George Mason University suggested a similar concept for DIS called the Selective Reliability Transport Protocol.) Previous game versions used TCP. New versions have a protocol that

- guarantees the reliability of certain packets with TCP—such as error conditions and session setup—and information sent infrequently (server message of the day);
- does not guarantee reliability for frequent and noncritical data such as player state (speed, direction);
- allows switching on demand from TCP to UDP (User Datagram Protocol)/TCP and back; and
- won't hang or cause abnormal termination if a UDP packet is lost.

Views

Views are windows into the VE from a person's or processes' perspective. We define two useful views for distributed environments: synchronous and asynchronous. An example synchronous view is in a distributed flight simulator, where one machine controls the forward image and two other hosts each process the left and right cockpit window perspectives. The images are coordinated to give the illusion that they are all part of a single cockpit view.

Synchronism demands both high reliability and low latency. Therefore, VEs requiring synchronous views are restricted to high-speed, local networks. An example of such an environment is the Raven simulator developed by Southwest Research Institute for NASA. It synchronizes shuttle astronaut viewpoints and renders them on different machines to improve rendering performance. The CAVE uses a similar approach to synchronize each image frame projected on a hemicube's screen. It also synchronizes the simulation run separately on a CM-5. Originally, this was done using a ScramNet (proprietary fiber optic, shared memory LAN). Later, it was accomplished using multiple raster managers on a Silicon Graphics Reality Engine Onyx and shared memory.

Synchronous views are also important for computer-aided design and systems used for concurrent engineering. The Fraunhofer Institute for Computer Graphics developed Coconut, a virtual prototyping environment that implements a shared 3D CAD viewer (Figure 6).[12] The VE demonstrates the ability of engineering teams to work globally via the Fraunhofer Transatlantic Research and Development Environment (Trade).

In the paradigm of the asynchronous view, multiple users have individual control over when and what they can see concurrently in the VE (Figure 7). Participants can be physically separated over a LAN or a WAN. Their awareness of each other's presence, if they are represented by an object, is brought about inside the VE. The Naval Postgraduate School's VE, NPSNet, supports the DIS protocol and uses the asynchronous model where each view is a simulated entity (http://www.cs.nps.navy.mil/research/npsnet). Views not associated with an entity are often referred to as "magic carpets" or "stealth" vehicles (Figure 8). Stealth entities just "listen" to the distributed world traffic because the world does not need to have knowledge of the viewer. Large-scale VEs will use asynchronous views because they are more cost efficient than WANs. Synchronous views will be important for small VEs requiring precise cooperative object manipulation and device communications distributed over LANs.

Data

Perhaps the most difficult decision in building a distributed environment is determining where to put the data relevant to the state of the virtual world and its objects. These decisions affect the VE data's scale, communication requirements, and reliability. For example, a real-time system demanding strong consistency will be inherently difficult to scale because it requires causality and automaticity. For now, at least, large VEs only allow weak consistency among group members. There are many conceivable ways of distributing persistent or semi-persistent data (see Figure 9). Here we present some of the methods most prevalent in current VEs.

Replicated homogeneous world. A common method for large VEs is to initialize the state of every system participating in the distributed environment with a homogeneous world database containing information about the terrain, model geometry, textures, and behavior of everything represented in the VE. Object state changes such as vehicle location or events such as the detonation of a simulated missile or collisions between two objects are communicated among all environment users. The advantage of this approach is that messages remain relatively small. The disadvantages are, it is relatively inflexible and as VE content increases, so must everyone's database. Moreover, over time, the world becomes inconsistent among the participants through the loss of state and event messages. This is the model for SimNet. However, once a simulation begins, each host maintains its own database without making any effort at guaranteeing consistency except through the use of "heartbeat" messages and event updates.

Image courtesy of Michael Zyda, Naval Postgraduate School

Figure 8. NPSNet's asynchronous view of "stealth" vehicles.

- Replicated homogeneous world database
  SimNet
- Shared, centralized
  Vistel
- Shared, distributed, peer-to-peer
  DIVE
- Shared, distributed, client-server
  BrickNet

Figure 9. Distributed data models for VEs.

Shared, centralized databases. On the other hand, the Virtual Space Teleconferencing System (Vistel) uses a shared world database.[13] As its name implies, Vistel is a teleconferencing system that displays 3D models of each conference participant. Changes in a model's shape, reflecting changes in a person's facial expression, are sent via messages to a central server and redistributed. Only one user at a time can modify the database (see Figure 10, next page).

MUDs use this model, although they employ relatively primitive clients. Text-based MUDs use TCP connections to a central server that does almost all the computation and maintains the virtual world's state. For text communication, this typically scales to about 50 concurrent users who "move" about among rooms, create and delete new objects or actions, and communicate with each other. LamdaMoo from Xerox Palo Alto Research Center is probably the most advanced MUD (ftp://ftp.parc.xerox.com/pub/MOO/papers/MUDsGrowUp.ps). Using a centralized server for 3D virtual worlds obviously limits participants to a few because of I/O

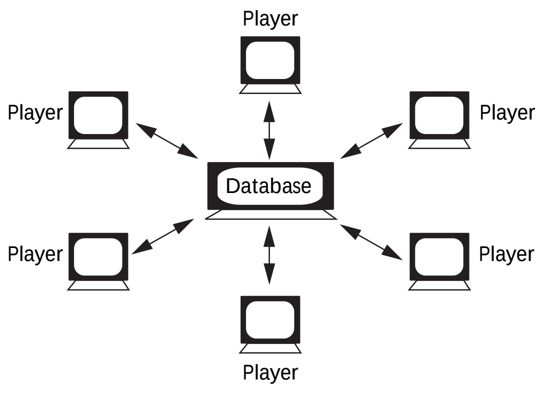


Figure 10. A centralized database model.

contention and the difficulty of maintaining a dynamic object database in real time. The I/O problem afflicts Netrek, which scales to about 18 players with UDP and uses an asymmetric communications model. The data in Figure 11 from J. Mark Noworolski shows how the server becomes the bottleneck because it must retransmit all other players' state to each client. In this case, communication from individual clients reaches only 168 bps. The server, however, must take every client message and redistribute it to all the other clients with a required order of magnitude increase in bandwidth.

```
Server -> Client network usage:
 Maximum CPS during normal play: 3,588 bps
Standard deviation: 918
Total bytes received 1,795,888, average CPS: 803.0

Client -> Server network usage:
 Maximum CPS out during normal play: 168 bps
Standard deviation out: 21
Total bytes sent 20,580, average CPS: 18.0
```

Figure 11. Server versus client communication in Netrek.

Shared, distributed databases with peer-to-peer updates. Many distributed systems strive to simulate shared memory architectures. For example, DIVE has a homogeneous, fully replicated, distributed database. However, unlike SimNet, the entire database is dynamic and uses reliable multicast protocols to actively replicate new objects. A disadvantage with this approach is the difficulty of scaling up because of the communications costs associated with maintaining reliability and consistent data across WANs. Modeling complex or dense objects, such as constructing a large CAD model or changing a terrain database, is very expensive (though highly desirable) in terms of the number of polygons that might be created, changed, and communicated over a network. VEs that use Linda, the parallel programming language, also trade performance for a relatively simple blackboard programming model. For example, Amselem[14] developed a VE using Linda with an unusual hand-held interface—a portable LCD television with a space tracker for VE navigation. The multiuser system's lackluster performance limited its use to three participants. Linda's simplicity and illusion of shared memory also contributed to the system's poor performance. Data must reside somewhere. In this case, it was on a central server.

Shared, distributed, client-server databases. Another technique uses a variant of the client-server model in which the database is partitioned among clients and a central server mediates communication. In WorldNet, as an entity moves through the VE, an object-request broker on a server that has knowledge of which client maintains that part of the world updates WorldNet's database.[15] WorldNet is more appropriate for large CAD or game environments because it tackles the walkthrough problem of a VE that has huge numbers of component models and provides multiple views simultaneously to user groups. However, in a dynamic large-scale world, the servers can quickly become I/O bottlenecks, increasing the VE's inherent latency.

In a similar approach to WorldNet and DIVE, the Model, Architecture, and System for Spatial Interaction in Virtual Environments[16] (Massive) system uses a spatial model for data partitioning among clients. In this case, an entity declares its world to a local "aura" manager, which in turn informs other aura collision managers. These managers broker between objects by detecting proximal collisions and informing each peer entity's mutual interface references. Pure client-server systems that strictly use classic remote procedure calls (RPCs) do not scale well for a number of reasons. RPCs are poorly suited for high-speed networks because communication depends on sending a message and waiting for a reply. As relative network delays increase, RPCs become expensive.

Processes

Distributing processes to multiple hosts increases a simulation's aggregate computing power. We can use this not only to distribute views but also to handle different input devices. SimNet and its descendants, such as DIS-compliant systems, use aggregate computing power by taking advantage of dead-reckoning to reduce the need for network communication.

With the exception of the DIS model, practically all distributed environments assume that similar processes run on each host that has the same function (although architectures may differ). The advantage of this approach is consistency. The disadvantage is inflexibility. The DIS protocol lets different developers create different simulations on different machines that theoretically can share in the same VE because they can communicate at some common level. Unfortunately, no protocol is complete, including DIS. For example, new objects cannot be introduced without a change in the standard.

The Aviary system has homogenous processes but contains object servers, which permits migration of lightweight objects to enable load balancing. These objects represent the entities and processes that control them. DIVE uses process groups from ISIS to partition the VE into rooms or spatial regions. The MR Toolkit distributes processes that support different VE components, such as input devices.[2] It provides an interpreted language, the Object Modeling Language (OML) that allows platform independence for developing VEs. OML specifies a VE entity's behavior and geometry. Similarly, WorldNet uses a language called Starship. WorldNet objects can share or transfer behaviors specified in Starship. These behaviors are either environmentally dependent, reactive, or capability based.

Other more general-purpose scripting languages use active messaging to migrate processes and objects across diverse platforms. Sun's Java is the primary example of this class. A byte-compiled, interpretive language, similar to C++, Java melds to the World Wide Web's client-server architecture. Java also supports peer IP Multicast communication and, in the future, a host of multimedia components.

The Virtual Reality Modeling Language (VRML) is a language for describing multi-participant interactive simulations—virtual worlds networked via the global Internet and hyperlinked with the World Wide Web (http://www.eit.com/vrml/vrmlspec.html). VRML describes 3D scenes and methods for interacting with models. Though VRML 2.0 lets Java provide object behaviors, VRML itself does not provide a mechanism for communication among distributed users.

Telescript is an interpreted language specifically designed for communication. It provides primitives that allow the script to suspend, migrate to another network node, and resume execution from the same point. The key idea is procedural messaging. With Telescript, write agents are sent around the network to accomplish the tasks you want. Instead of having a client dealing with a server by sending messages back and forth, you build an agent and send it where the server resides. The agent is smart enough to interact with the server and returns the required information to the sender. This reduces bandwidth consumption and lets users build agents that seek information on our behalf.

In a distributed VE, clients can be homogeneous, as in DIVE. Clients can also be dissimilar except for the communications protocols among them, providing interoperability (for example, DIS and SimNet, which exchange standard state and event data). Furthermore, processes can migrate across homogeneous architectures like Aviary. New scripting languages like Safe-Tcl offer the opportunity for migrating processes across heterogeneous systems, therefore efficiently exchanging object behaviors as well as entity states within large, heterogenous VEs.

Summary

We have discussed VEs in the context of how communications, views, data, and processes are distributed. We have not exhausted all the considerations for developing VEs but have emphasized those aspects critical to scaling environments. Most of the systems described here scale to accommodate a handful of users. We also know that systems that demand strong data consistency, causality, and reliable communications, and at the same time need to support real-time interaction, are not likely to scale well. Geographically dispersed systems require high-speed, multicast communication. Replicated world databases are more communication efficient than centralized or distributed shared database schemes. However, they generally lack a way to maintain world consistency—a problem with unreliable transport mechanisms like UDP. They also cannot update the VE with new objects or behaviors. However, large VEs could use a mixed model—client initialization with small replicated data sets and a distributed client-server model. This would allow more data consistency and persistence if a mechanism or heuristic is used to reduce transfer latency.

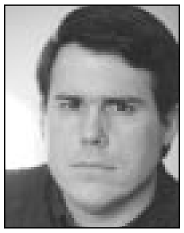
We believe that the current trend toward a pure client-server paradigm of the WWW in VEs is a future dead-end unless we use hybrid architectures that incorporate peer communications. New languages like Java and VRML will provide innovative methods for building virtual worlds. However, the problem of achieving scalability, reliability, and real-time interaction simultaneously will likely not be resolved soon. MM

Acknowledgments

A grateful thanks to Rich Gossweiller for letting us use his figures (Figures 3, 7, and 10). This work would not have been possible without the support of our research sponsors: Darpa, DMSO, USA Stricom, USA TRAC. A version of this article was originally presented at the Second IEEE Workshop on Networked Realities, Boston, Massachusetts, 26-28 October, 1995.

References

1. T.M. Roy et al., “Steering a High Performance Computing Application from a Virtual Environment,” *Presence*, Vol. 4, No. 2, Summer 1995, pp. 110-120.
2. C. Shaw and M. Green, “The MR Toolkit Peers Package and Experiment,” *Proc. VRAIS 93,* IEEE Press, Piscataway, N.J., 1993, pp. 463-469.
3. J.W. Barrus, R.C. Waters, and D.B. Anderson, “Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments,” *Proc. VRAIS 96*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1996, pp. 204–213.
4. B.N. Schilit and M.M. Theimer, “Disseminating Active Map Information to Mobile Hosts,” *IEEE Network*, Sept. 1994, pp. 22-32.
5. S. Benford et al., “Managing Mutual Awareness in Collaborative Virtual Environments,” *Proc. VRST 94*, World Scientific Publishing, N.J., 1994, pp. 223-236.
6. C. Carlsson and O. Hagsand, “DIVE—A Multiuser Virtual Reality System,” *Proc. VRAIS 93*, IEEE Press, Piscataway, N.J., 1993, pp. 394-400.
7. D.J. Van Hook and J.O. Calvin, “Approaches to Relevance Filtering,” *11th DIS Workshop*, Inst. For Simulation and Training, Orlando, Fla., Sept. 1994, pp. 367-369.
8. M. Pullen, “Dual-Mode Multicast,” DIS Communications Architecture Subgroup Winter Workshop, Jan. 1995, http://bacon.gmu.edu/lsma/publications.html.
9. D. Miller, “Long-Haul Networking of Simulators,” *Proc. Tenth Interservice/Industry Training Systems Conf.*, ADPA, Arlington, Va., 1989, p. 2.
10. S.K. Singhal, “Using a Position History-Based Protocol for Distributed Object Visualization,”*Designing Real-Time Graphics for Entertainment* (Course Notes for Siggraph 94 Course #14), ACM Press, N.Y., July 1994.
11. C. Sayers and R. Paul, “An Operator Interface for Teleprogramming Employing Synthetic Fixtures,” *Presence*, Vol. 3, No. 4, Winter 1994, pp. 309-320.
12. B.G. Anderson, U. Jasnoch, and H. Joseph, “Coconut—A Virtual Protoyping Environment,” *Computer Graphic Topics*, Vol. 8, Mar. 1996, pp. 20-22.
13. J. Ohya et al., “Real-Time Reproduction of 3D Human Images in Virtual Space Teleconferencing,” *Proc. VRAIS 93*, IEEE Press, Piscataway, N.J., 1993, pp. 408-414.
14. D. Amselem, “A Window on Shared Virtual Environments,” *Presence*, Vol. 4, No. 2, Spring 1995, pp. 140-145.
15. G. Singh, “A Software Toolkit for Network-Based Virtual Environments,” *Presence*, Vol. 3, No. 1, Summer 1994, pp. 19-34.
16. C. Greenhalgh and S. Benford, “Massive, A Collaborative Virtual Environment for Tele-Conferencing,” *ACM Trans. Computer-Human Interaction* (special issue on Virtual Reality Software and Technology), Vol. 2, No. 3, 1995, pp. 239-261.



Michael R. Macedonia is vice president in charge of developing global work environments for commerce, government, and education at the nonprofit Fraunhofer Center for Research in Computer Graphics, in Providence, Rhode Island. He received a BS from the US Military Academy in 1979, an MS in telecommunications from the University of Pittsburgh in 1989, and a PhD in computer science from the Naval Postgraduate School in 1995. He is a member of the IEEE Computer Society, IEEE Communications Society, ACM Siggraph and Sigcomm, Society for Computer Simulation, and Society of Manufacturing Engineers.

Michael Zyda is a professor and the Academic Chair in the Department of Computer Science at the Naval Postgraduate School, Monterey, California. He received a BA in bioengineering from the University of California, San Diego in 1976, an MS in computer science/neurocybernetics from the University of Massachusetts, Amherst in 1978, and a DSc in computer science from Washington University, St. Louis, Missouri in 1984. His main research focus is in computer graphics, specifically the development of large-scale, networked 3D virtual environments. Zyda is a member of the Technical Advisory Board of the Fraunhofer Center for Research in Computer Graphics, Providence, Rhode Island.

Contact Macedonia at Fraunhofer Center for Research in Computer Graphics, 321 S. Main Street, Providence, RI, 02906, e-mail mmacedon@crcg.edu, Web site http://www.crcg.edu.