Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996
Location: Santa Clara, CA , USA
30 Mar-3 Apr 1996, pp. 222-228.

Network Topologies for Scalable Multi-User Virtual Environments

Thomas A. Funkhouser
Bell Laboratories
600 Mountain Avenue, 2A-202
Murray Hill, NJ 07974
funk@research.att.com

Abstract

This paper investigates trade-offs of different network topologies and messaging protocols for multi-user-virtual environment systems. We present message distribution techniques appropriate for constructing scalable multi-user systems for a variety of network characteristics. Hierarchical system designs utilizing servers that manage message distribution for entities in separate regions of a virtual environment are described that scale to arbitrary numbers of simultaneous users. Experimental results show that the rate of messages processed by server workstations in this system design are less than using previously described approaches.

1. Introduction

With the recent increases in network bandwidth and graphics performance in desktop computers, there is a growing interest in distributed visual simulation systems that allow multiple users to interact in a shared 3D virtual environment. Users on workstations connnected by a wide-area network run an interactive 3D graphics interface program that simulates the experience of immersion in a virtual environment by rendering images of the environment as perceived from the user's simulated viewpoint. As each user is represented in the shared virtual environment by an entity (avatar) whose state is controlled by user input and kept up-to-date on participating workstations via messages, these systems support visual interactions between multiple users in a shared 3D virtual environment. Applications for this technology include distributed training simulations, collaborative design, virtual meetings, and multiplayer games.

A difficult challenge in multi-user visual simulation is maintaining consistent state among a large number of workstations distributed over a wide-area network. Whenever any entity changes state (e.g., moves) or modifies the shared environment, an appropriate update must be applied on every workstation participating in the shared virtual environment. If N entities move through a shared virtual environment simultaneously, each modifying its position and/or orientation M times per second, then $M*N$ updates are generated to a shared database per second.

For a given set of network characteristics, the design of a multi-user virtual environment system may have dramatic impact on the system's scalability and message distribution performance. Some system designs may scale to many simultaneous users, while others may lead to saturation of a particular workstation or network connection with relatively few users. Moreover, a system design appropriate for one set of network characteristics may be inappropriate for another. The goal of this paper is to investigate the trade-offs of different network topologies and messaging protocols for multi-user-virtual environments. Our aim is to characterize the design of systems that can scale to very large numbers of simultaneous users.

The rest of the paper is organized as follows. The next section contains a summary of related work. Section 3 describes some network characteristics that impact the message distribution properties of multi-user virtual environment systems. Several possible system designs based on different network topologies are described in Section 4. Section 5 presents quantitative results from experiments with two hierarchical system designs, and discusses implications of these results for scalable multi-user virtual environments. Finally, a brief summary and conclusion appear in Section 6.

0-8186-7295-1/96 $5.00 © 1996 IEEE
Proceedings of VRAIS '96

2. Related Work

Numerous systems have been developed for multi-user interaction in shared virtual environments. In general, these systems represent a virtual environment as a set of independent *entities* each of which has a geometric description and a behavior. Some entities are static (e.g., terrain, buildings, etc.), whereas others have dynamic behavior that can be either autonomous (e.g., robots) or controlled by a user via input devices (e.g., vehicles). Distributed simulation occurs when multiple entities interact in a shared virtual environment by sending messages to one another to announce updates to their own geometry or behavior, modifications to the shared environment, or impact on other entities.

Every entity is managed by one workstation participating in the distributed system. The workstation may map user input to control of particular entities and may include viewing capabilities in which the virtual environment is displayed on the client workstation screen from the point of view of one or more of its entities. In addition to managing its own entities (local entities), each workstation maintains surrogates for some entities managed by other workstations (remote entities). Surrogates contain (often simplified) representations for the entity's geometry and behavior. When a workstation receives an update message for a remote entity, it updates the geometric and behavioral models for the entity's local surrogate. Between updates, surrogate behavior is simulated by every workstation.

Multi-user virtual environment systems can be characterized by their approach to message distribution. For instance, Reality Built For Two [2], VEOS [4], and MR Toolkit [12] are based on unicast peer-to-peer designs. A unicast message is sent to each of N-1 workstations whenever any entity in the distributed simulation changes state. This approach yields $O(N^2)$ update messages during every simulation step, and thus does not scale to many simultaneous users before the network gets saturated.

SIMNET [5] and VERN [3] are also peer-to-peer systems, but use broadcast messages to send updates to all other workstations participating in a virtual environment at once. Although, this approach cuts down on the total number of messages transmitted to $O(N)$, every workstation still must process a message whenever any entity in the distributed simulation changes state. Since every workstation must store data and process update messages and/or simulate behavior for all N entities during every simulation step, these systems do not scale beyond the capabilities of the least powerful participating workstation.

NPSNET [16] and DIVE [6] are peer-to-peer systems that use multicast to send update messages to a subset of participating workstations. The general idea is to map entity properties into multicast groups, and send update messages only to relevant groups. For instance, NPSNET [11] partitions a virtual world into a 2D grid of hexagonal shaped cells each of which is represented by a separate multicast group. Entities localize their visual interactions by sending updates only to the multicast group representing the cell in which they reside, and they listen only to multicast groups representing cells within some radius. This approach scales well for many users, but is only practical for networks which allow peer-to-peer multicast messaging, and for mappings from entity attributes to multicast addresses that are relatively static so that the impact of messages and delays associated with joining and leaving multicast groups is minimal.

WAVES [9], BrickNet [13], and RING [8] are client-server systems. Communication between *client* workstations is managed by *message servers*. Clients do not send messages directly to other clients, but instead send them to servers which forward them to other clients and servers participating in the same distributed simulation. A key feature of the client-server design is that servers can process messages before propagating them to other clients, culling, augmenting, or altering the messages. For instance, a server may determine that a particular update message is relevant only to a small subset of clients and then propagate the message only to those clients or their servers. These systems scale well to many simultaneous users with intelligent server message processing.

The study presented in this paper is based on RING [8]. The initial version of RING used a "static" client-server design in which each client sent all its update messages to the same server. The system supported multiple inter-networked servers, but a single client was connected to the same server throughout its entire execution. The primary motivations for this design were to: 1) support modem connections to clients, and 2) simplify implementation. Although modems are an important class of network connections for client-server messaging, wide-area networks supporting connectionless unicast protocols (e.g., internet) and multicast protocols (e.g., MBONE) are available for many computers. If we construct a system using these networks, a client can send a message to any one or set of clients and/or servers at any time. As a result, a variety of alternate system topologies are possible, potentially with advantageous processing and messaging properties. Our aim is to investigate trade-offs for message distribution of multi-user virtual environment systems with alternate networking topologies.

3. Network Characteristics

Communication between workstations participating in a multi-user virtual environment can be implemented using a variety of possible networks with different characteristics. These logical networks can be classified by: 1) whether transport is connection-oriented or connectionless, 2) whether message delivery is unicast or multicast, 3) message latency, and 4) data bandwidth. In this paper, we consider wide-area networks of the following types:

- **Connection:** Two workstations can send data back and forth over a connection-oriented link. A primary example of such a network is a modem using a standard telephone line. A modem link supports two-way, connection-oriented, unicast data transport with relatively low latency and low bandwidth (14.4Kb/s or 28.8 Kb/s).
- **Unicast:** An arbitrary number of workstations are logically connected to a network supporting connectionless, unicast messages. The Internet can be used as a wide-area unicast network.
- **Multicast:** An arbitrary number of workstations communicate with each other with connectionless, multicast messages as well as connection-less, unicast messages. The MBONE is an example of a wide-area multicast network.

Heterogeneous networks can be constructed using combinations of different types of networks. For instance, modems might be used for connections between clients and servers, while servers communicate amongst themselves over the Internet. Each combination of networks has a unique set of transport modes and performance characteristics which can significantly impact design of systems for multi-user virtual environments.

4. Network Topologies

In this section, we present several possible network topologies for multi-user virtual environments and describe practical system designs for a variety of network characteristics.

Since a primary goal of this study is to investigate multi-user virtual environments for large numbers of users, we only consider system designs that scale – i.e., no single workstation can process all messages from all entities. All designs employ message filters based on precomputed line-of-sight visibility information in order to localize visual interactions (as used in RING [8]). Specifically, prior to the multi-user simulation, the shared virtual environment is partitioned into a spatial subdivision of *cells* whose boundaries are comprised of the static, axis-aligned polygons of the virtual environment [1, 14]. A visibility precomputation is performed in which the set of cells potentially visible to each cell is determined by tracing beams of possible sight-lines through transparent cell boundaries [14, 15]. During the multi-user visual simulation, real-time update messages are propagated only to the subset of workstations managing entities inside some cell visible to the one containing the updated entity.

Peer-to-Peer Topologies

In a peer-to-peer system design, the system is arranged with a set of workstations communicating over a network in which every peer can send messages directly to any other peer. If only a network supporting unicast messages is available, peers send a unicast message to other peers when an entity is updated (see Figure 1).

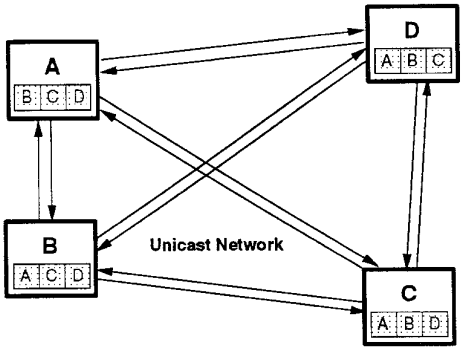

Figure 1. Peer-to-peer topology with a unicast network.

In order to scale the system to many simultaneous users, message filters must be applied so that update messages are not sent to every peer for every update. Peers can maintain lists of the entities resident in each cell and use precomputed cell-to-cell visibility information to send update messages only to other peers managing an entity residing in a cell visible to the cell in which an update occurred. This design scales beyond the simple $O(N^2)$ "send an update to everybody" approach as each peer receives only a subset of all update messages. However, it does not scale infinitely since all peers must maintain an up-to-date mapping of which entities are in each cell. In order to keep this mapping synchronized among peers, update messages are sent to all peers whenever an entity moves into a new cell

(hopefully, this is infrequently). The number of these "periodic" update messages may be relatively small, but it grows with $O(NP)$ for N entities and P peers.

If a network supporting multicast messages is available, peers can send a single multicast message to a subset of peers all at once (see Figure 2). A multicast group can be assigned to each cell. For each update, a peer sends a message to the multicast group representing the cell in which the update occurred, while all peers listen to the multicast groups representing the cells visible to the cells containing their entities [11]. With this approach, peers do not maintain explicit lists of entities resident in each cell; but, instead, they join and leave multicast groups as their entities move between cells. Filtering of messages is performed by the network rather than by the peers. Although peers do not exchange explicit "periodic" messages when entities move between cells, they join and leave multicast groups, which cause implicit messages to be generated by the multicast network to update routing tables. Since the number of multicast membership changes grows with $O(N)$, this design does not scale infinitely.

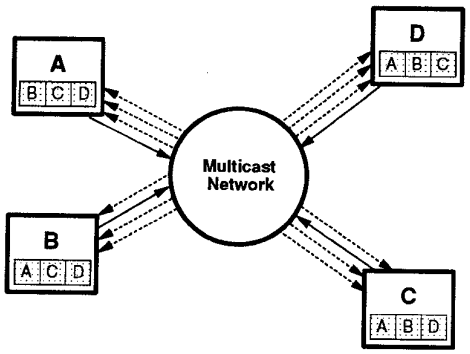

Figure 2. Peer-to-peer topology with a multicast network.

Hierarchical Topologies

Multi-user virtual environment systems can also be designed with a hierarchical topology in which message servers manage communication for their clients (see Figure 3). For each entity update, a client sends one update message to a server, and the server propagates the message to other servers and clients containing entities inside some cell visible to the one containing the updated entity [8]. The primary advantage of this approach is that the message distribution burden is shifted out of the clients and into servers. Since clients simply send one message to a server for each local update and receive messages from a server for updates to all relevant remote entities, they must perform very little processing, storage, or messaging to maintain consistent state among many entities in a large virtual environment. Client processing, storage, and network bandwidth requirements scale infinitely – i.e., they grow with density of entities, rather than the total number of entities in the virtual environment. The disadvantage of the hierarchical approach is that extra latency may be introduced for each update message.

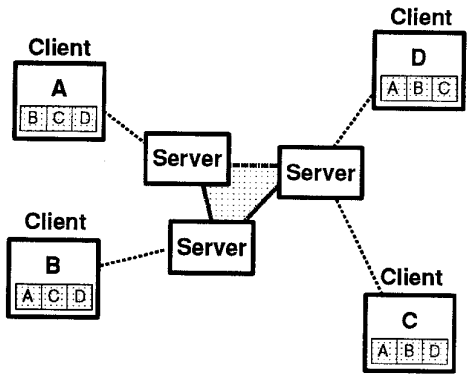

Figure 3. Hierarchical topology with client-server connections.

The message server processing requirements in a system with a hierarchical topology depend on the types of networks used for client-server and for server-server links.

If clients can communicate with only one server over a connection-oriented, unicast network (e.g., modem), then each server manages message distribution for a subset of the clients (this design was used in the original RING system – it is labeled 'A' in the next section). For every update to an entity, a client sends an update message to its server, and its server propagates the message to other servers and clients with entities inside some cell visible to the one containing the updated entity. In order to implement this approach, servers must maintain mappings of which entities are in each cell and exchange "periodic" update messages whenever an entity moves between cells. Therefore, this system design does not scale infinitely. However, the impact of message filtering in this hierarchical topology is far less than in the unicast peer-to-peer topology since: 1) there are fewer servers than peers (fewer unicast messages are required to update all), 2) the servers may have more available processing power (they do not render images or simulate entity behavior), 3) the servers may have more available memory (they do

not store display data for polygons or textures), and 4) the servers may be connected by faster networks.

If clients can communicate with any server using a connectionless, unicast network (e.g., Internet), then servers can be assigned to manage message distribution for separate regions of the virtual environment (this design is used in the current RING system – it is labeled 'B' in the next section). For each entity update, the client sends a unicast message to the server managing the region in which the update occurred. As always, servers propagate messages to other servers and clients with entities inside some cell visible to the one containing the updated entity. The advantage of this approach is that fewer server-to-server messages are generated, as most entity-entity visual interactions will occur between entities managed by the same server. As a result, far fewer update messages must be passed between servers in real-time. Likewise, each server must maintain mappings of which entities are in each cell only for the cells visible to its region, so periodic update messages must be sent only between servers whose regions are potentially visible to each other. This design scales infinitely as the regions of interest of both clients and servers are limited to finite subsets of the virtual environment.

If servers can communicate with each other over a multicast network (.e.g, MBONE), then they can distribute messages using an approach similar to the one used for the multicast peer-to-peer system design described in the previous section. For each entity update, the client sends a message to a server, which relays the message to the multicast group representing the cell in which the update occurred. Each server listens only to the multicast groups for the cells visible to one region of the virtual environment, and maintains lists of which entities are in each cell only for cells in that region. When a server receives a multicast message, it propagates it to clients with entities residing in the cell represented by the multicast group. As the region managed by each server is static (or at least changing very infrequently), servers do not join and leave multicast groups dynamically as is required in a peer-to-peer multicast system. As a result, no "periodic" update messages are required when entities move between cells, and this system design scales infinitely.

Of course, we may also organize a system in a multi-level hierarchy with second level message servers managing communication between first level servers. Which organizations are most efficient depends on the characteristics of underlying networks.

5. Experimental Results

An experimental system has been implemented for study of message distribution for multi-user virtual environments. The system runs on Silicon Graphics workstations and uses UDP datagrams for message passing.

We ran experiments with this system in a virtual environment with 800 "rooms" connected by "hallways" consisting of 23,168 polygons and 2,219 cells (one tile is shown in Figure 4). All experiments were run with 256 computer-controlled entities simultaneously navigating through the virtual environment "randomly," following piecewise linear paths in randomized directions for randomized distances. During these experiments, clients sent update messages only for changes in derivatives of entity position and/or orientation (i.e., dead-reckoning) while other clients simulated intermediate positions with linear "smooth-back." Update messages containing 40 bytes (message-type[4], entity-ID[4], target-position[12], target-orientation[12], positional-velocity[4], and rotational-velocity[4]) were generated for each entity once every 2.3 seconds on average with this "random" navigational behavior.

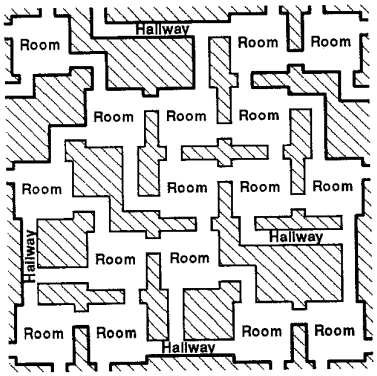

Figure 4. Test virtual environment is 8x4 repeating tile pattern of rooms connected by hallways.

To demonstrate the effect of system design on the message processing requirements of workstations in a multi-user virtual environment, we performed experiments using two of the hierarchical system designs described in the previous section: A) clients made static connections to one server, while servers passed messages to each other using a connection-less, unicast network, and B) clients and servers both passed messages on a connection-less, unicast network – each server managed message distribution for a separate region of the virtual environment. For each of these two

System Design	Number of Servers	Client		Server					
		Input	Output	Input			Output		
		From Server	To Server	From Client	From Server	Total	To Client	To Server	Total
Static (A)	2	4.25	0.50	46.8	44.0	90.8	290.5	44.0	334.5
	4	5.09	0.46	24.1	64.4	88.5	148.6	64.7	213.3
	8	3.29	0.45	12.0	70.4	82.3	71.3	71.5	142.8
	16	4.16	0.37	6.1	74.7	80.8	37.2	75.8	113.0
Regional (B)	2	4.67	0.42	48.2	1.9	50.2	270.7	2.0	272.7
	4	3.89	0.53	24.2	4.3	28.5	153.0	4.3	157.3
	8	4.15	0.46	12.3	3.6	15.8	83.5	3.6	87.1
	16	4.41	0.43	5.9	3.4	9.3	37.5	3.5	41.0

Table 1. Average message processing rates (messages per second) measured in a single server during tests with 2, 4, 8, and 16 servers using A) static connections between clients and servers, and B) connection-less networks between clients and regional servers.

system designs, we logged counts of input and output messages in each client and server during tests with 2, 4, 8, and 16 servers. Table 1 lists average client input/output and server input/output message rates during these tests.

In system design (A), clients were connected to servers statically since connection-oriented networks were used for client-server links. Servers processed messages for the subset of the entities visible to the entities on those clients. Since the entities managed by any client were generally spread evenly through the virtual environment, a large percentage of update messages were visible to some entity managed by each server, and consequently a large number of messages were passed between servers (see column 6 of Table 1). As the number of servers increased, the total number of server-server messages output by each server increased as separate unicast messages were sent to multiple servers. This increase was sub-linear, however, due to the visibility-based filtering of messages between servers.

In system design (B), connection-less networks were used for client-server links, and clients sent update messages to different servers based on the region of the virtual environment in which the update occurred. As a result, each server processed messages for the subset of the entities visible to a separate region of the virtual environment. Since most visual interactions occurred between entities managed by the same server, very little server-server messaging was required. As the number of servers increased, and the inter-visibility between server regions decreased, server-server messaging was reduced (see column 6 of Table 1). Using this system design, the message processing burden of each client and server can be quite small, and these systems scale to many simultaneous users.

6. Conclusion

Characteristics of the networks used to construct multi-user virtual environment systems can greatly impact the message distribution performance of a particular system design. For instance, availability of connectionless messaging protocols allows construction of infinitely scalable systems. Support for multicast protocols allow portions of the message distribution processing to be performed by the network routers rather than by peer or server workstations. Economical considerations may also impact system design. For example, it may be more affordable to build large systems using a hierarchical topology in which message distribution is off-loaded from many low-cost client workstations into a relatively few server workstations.

The results of experiments presented in this paper demonstrate that different network characteristics and different system designs can significantly affect the message processing rates required by workstations in a multi-user virtual environment. We found that a hierarchical system design in which clients send messages to regional servers via connection-less unicast networks scales better than a hierarchical design in which clients make static connections to one server. Perhaps, by identifying network characteristics and system designs that improve the message distribution properites (or decrease the cost) of multi-user virtual environment systems, we can aid software and network architects in the design of future systems.

References

[1] Airey, John M., John H. Rohlf, and Frederick P. Brooks, Jr., Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. *ACM SIGGRAPH Special Issue on 1990 Symposium on Interactive 3D Graphics*, 24, 2 (1990), 41-50.

[2] Blanchard, C., S. Gurgess, Y. Harvill, J. Lanier, A. Lasko, M. Oberman, and M. Teitel, Reality Built for Two: A Virtual Reality Tool. *ACM SIGGRAPH Special Issue on 1990 Symposium on Interactive 3D Graphics*, (Snowbird, Utah), 1990, 35-36.

[3] Blau, Brian, Charles E. Hughes, Michael J. Moshell, and Curtis Lisle, Networked Virtual Environments. *ACM SIGGRAPH Special Issue on 1992 Symposium on Interactive 3D Graphics*, (Cambridge, MA), 1992, 157-164.

[4] Bricken, William, and Geoffrey Coco *The VEOS Project.* Technical Report, Human Interface Technology Laboratory, University of Washington, 1993.

[5] Calvin, James, Alan Dickens, Bob Gaines, Paul Metzger, Dale Miller, and Dan Owen, The SIMNET Virtual World Architecture. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, September, 1993, 450-455.

[6] Carlsson, Christer, and Olof Hafsand, Dive: A Multi-User Virtual Reality System. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, September, 1993, 394-401.

[7] Funkhouser, Thomas A., Carlo H. Séquin, and Seth J. Teller, Management of Large Amounts of Data in Interactive Building Walkthroughs. *ACM SIGGRAPH Special Issue on 1992 Symposium on Interactive 3D Graphics*, (Cambridge, MA), 1992, 11-20.

[8] Funkhouser, Thomas A. RING: A Client-Server System for Multi-User Virtual Environments. *ACM SIGGRAPH Special Issue on 1995 Symposium on Interactive 3D Graphics*, (Monterey, CA), 1995, 85-92.

[9] Kazman, Rick, Making WAVES: On the Design of Architectures for Low-end Distributed Virtual Environments. *Proceedings of IEEE Virtual Reality Annual International Symposium*, September 1993, 443-449.

[10] Kazman, Rick, Load Balancing, Latency Management and Separation of Concerns in a Distributed Virtual World. *Parallel Computations - Paradigms and Applications*, A. Zomaya (ed.), Chapman & Hall, 1995, to appear.

[11] Macedonia, Michael, R. Michael J. Zyda, David R. Pratt, and Paul T Barham, Exploiting Reality with Multicast Groups: A Network Architecture for Large Scale Virtual Environments. To appear in *Proceedings of IEEE Virtual Reality Annual International Symposium*, 1995.

[12] Shaw, Chris, and Mark Green, The MR Toolkit Peers Package and Experiment. *Proceedings of IEEE Virtual Reality Annual International Symposium*, September 1993, 463-469.

[13] Singh, Gurminder, Luis Serra, Willie Png, Audrey Wong, and Hern Ng, BrickNet: Sharing Object Behaviors on the Net. *Proceedings of IEEE Virtual Reality Annual International Symposium*, March, 1995, 19-25.

[14] Teller, Seth J., and Carlo H. Séquin, Visibility Preprocessing for Interactive Walkthroughs. *Computer Graphics* (SIGGRAPH '91). 25, 4, 61-69.

[15] Teller, Seth J., *Visibility Computations in Densely Occluded Polyhedral Environments.* Ph.D. thesis, Computer Science Division (EECS), University of California, Berkeley, 1992. Also available as UC Berkeley technical report UCB/CSD-92-708.

[16] Zyda, Michael J., David R. Pratt, John S. Falby, Chuck Lombardo, and Kristen M. Kelleher, The Software Required for the Computer Generation of Virtual Environments. *Presence*, 2, 2 (March 1993), 130-140.