The Origin of Networked Virtual Environments

448430
Spring 2009
3/16/2009
Kyoung Shin Park
Multimedia Engineering
Dankook University

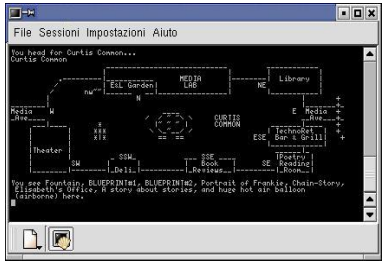
Networked Virtual Environments

- Text-based Online Community
  - MUDs & MOOs
- Networked Games
  - SGI Flight & Dogfight
  - DOOM
  - MMORPG
- Online Community
  - Active World, Blaxxun, Community Place, Open Community
  - Second Life
- Networked Virtual Environments
  - SIMNET/Distributed Interactive Simulation (DIS)/NPSNET
  - PARADISE
  - DIVE
  - BrickNet
  - MR Toolkit
  - Diamond Park
  - CAVERNsoft

Games & Communities

MUDs & MOOs

- MUD (Multi-User Domain) by Richard Bartle and Roy Trubshaw in 1978
- MOO (MUD, Object-Oriented)
- Text-based VR environment
- Originally designed as a form of the Dungeons and Dragons game
- Developed for multi-users
- Allows users to interact both with their environment and with other users
- Descriptions of real and imagined areas such as forests, dungeons, offices, universities, cities, rooms, or any other spatially oriented environment
- Communication commands are modeled on real life, with “say”, “tell”, “whisper” and “shout”



SGI Flight & Dogfight

- SGI Flight
  - 3D aero plane simulator demo for Silicon Graphics workstation in 1983-1984.
  - Gary Tarolli of Silicon Graphics, Inc. is probably the person that most in the networked virtual environment community would credit as the originator of their thoughts on NVEs.
  - Users could see each other's plane but no interaction.
  - Networking was added into Flight in stages, beginning in 1984.
  - The initial networked version of Flight actually used a serial cable between two SGI workstations and ran at something like 7 frames per second on a Motorola 68000 based workstation (about 1 MIPS with maybe 500 polygons per second graphics capability).
  - Flight was distributed in networked form on all SGI workstations sometime after SIGGRAPH 1984 and could be seen in practically every SGI-outfitted lab at that time, either during the day on breaks or after hours.

SGI Flight & Dogfight

- SGI Dogfight
  - Sometime after the release of the networked version of Flight, in early 1985 it is believed, SGI engineers modified the code of Flight to produce the demonstration program Dogfight.
  - This modification dramatically upgraded the visibility of NVEs as players could now interact by shooting at each other.
  - It did not help that Dogfight packets were transmitted at frame rate and were clogging the network.
  - Flight/Dogfight inspired many to develop their own NVE.
  - We believe that networked Flight/Dogfight inspired the development of more networked VEs and games than SIMNET and DIS combined.
  - SGI made the source code to Flight and Dogfight freely available and many people asked for the code just so they could learn how to read and write UDP packets.

Doom

- On Dec 1993, id Software released its shareware game, Doom.
- First-person shooter (FPS) for PCs
- The shareware giveaway of the first level of Doom is probably singularly responsible for the rush of startups into the business of providing online gaming networks.
- This networked ability to blast people in a believable 3D environment created enormous demand for further 3D networked games.

Doom

- The posting of Doom caught most network administrators' eyes when their LANs started bogging down. Doom did no dead reckoning and flooded LANs with packets at frame rate.
- This networked ability to blast people in a believable 3D environment created enormous demand for further 3D networked games.
- An estimated 15 million shareware copies of Doom have been downloaded around the world, passed from player to player by floppy disk or online networks.

Online Community

- Internet-based networked virtual environments will impact the greatest number of people.
- Examples are Active World, Blaxxun, Sony's Community Place, Open Community, Vnet.
- Problems of latency, rendering, inconsistency, lack of interaction

Online Community

'Dokdo is Korea Territory!' in Second Life
www.serakorea.com

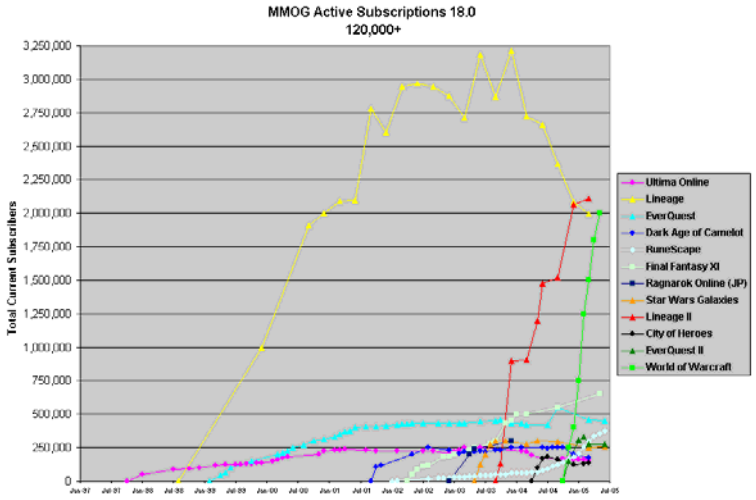
- Second Life is an online 3D virtual world community, developed by Linden Lab and modelled after the Metaverse of Snow Crash.

Online Game

Ultima Online

- Massively multiplayer online role-playing game (MMORPG) is a genre of computer role-playing games, in which a large number of players interact with one another in a virtual world.
- Richard Garriott, the creator of Ultima Online, coined the term MMORPG.
- Popular examples are Neverwinter Nights, Ultima Online, EverQuest, Blizzard's World of Warcraft.

Online Game



(1997 - 2005)

From, Shawn Rider "MMORPGs: MUD to Wow" (2006/3/7)

Department of Defense
Networked Virtual Environments

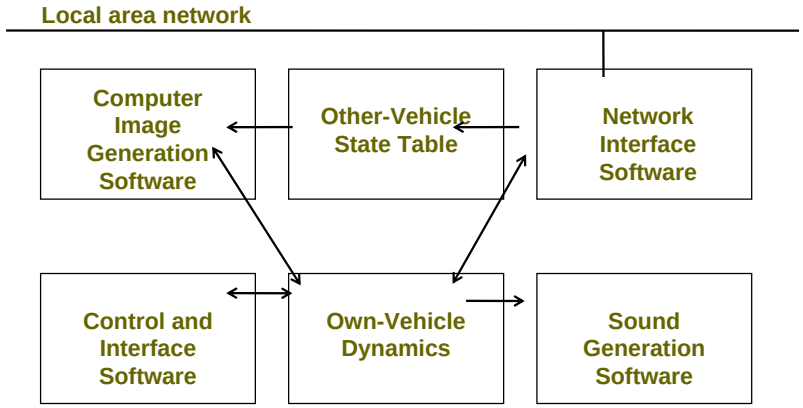
SIMNET (Simulator Networking)

- SIMNET (simulator networking) is a distributed military virtual environment for DARPA, begun in 1983, delivered to the U.S. Army at the end of March 1990.
- The goal was to develop a "low-cost" networked virtual environment for training small units to fight as a team.
- SIMNET project created an 11-site testbed with from 50 to 100 simulators at each site.

SIMNET

- The key technical challenges for SIMNET were
  - How to fabricate high quality, low-cost simulators and
  - How to network them together to create a consistent, virtual battlefield.
- Testbed
  - To carry out the study of these technical challenges, the SIMNET project created an 11 site testbed with from 50 to 100 simulators at each site.
  - SIMNET could be entered from anywhere on the network using a simulator as the portal into the synthetic environment. Once that synthetic environment was entered, the "player" could interact with others who were also online in the synthetic battlefield.
  - Play in that synthetic environment was unscripted, basically free-play, done within the confines of whatever chain of command was imposed on the simulation participants.

SIMNET Architecture



SIMNET Network Software Architecture

- The SIMNET network software architecture has three basic components:
  - An object-event architecture,
  - A notion of autonomous simulation nodes, and
  - An embedded set of predictive modeling algorithms called "dead reckoning"

SIMNET Network Software Architecture

- The SIMNET object-event architecture modeled the world as a collection of objects whose interactions with each other are just a collection of events.
  - Objects are the vehicles and weapons systems that can interact across the network.
  - Events in SIMNET are messages to the network indicating a change in world or object state.
  - In SIMNET, the notion of autonomous simulation nodes means that individual players (vehicles and weapons systems) on the network are responsible for placing messages, or packets, onto the network to accurately represent their current state.
  - Packet recipients are responsible for receiving such state change information and making the appropriate changes to their local model of the world.

SIMNET

- Autonomous Simulator Nodes
  - In SIMNET, the notion of autonomous simulation nodes means that individual players (vehicles and weapons systems) on the network are responsible for placing messages, or packets, onto the network to accurately represent their current state.
  - Packet recipients are responsible for receiving such state change information and making the appropriate changes to their local model of the world.
  - This lack of a central server means that single point failures in the system do not take the entire simulation down. Additionally, it allows players to join and leave the simulation at any time.
  - Placing changes in state on the network means that a node must place packets onto the network whenever its objects have changed enough so that the other players should be made cognizant of the changes.
  - Placing current state onto the network also means that a node must provide a regular "heartbeat" message, usually every 5 seconds, to keep other players informed that a particular object is alive and still in the system (and hence should be displayed).

SIMNET

- Dead-Reckoning
  - To reduce packet traffic in SIMNET, the objects and ghosts paradigm was created. The idea behind this paradigm is that objects only place packets onto the network when their home node determines that the other nodes on the network are no longer able to predict their state within a certain threshold amount.
  - This paradigm assumes that the other nodes in the system are maintaining "ghost" copies of the object in their memories and that the last reported direction, velocity, and position are sufficient to predict, within the threshold amount, where that entity is now.

SIMNET Scalability

- SIMNET Scalability
  - The SIMNET network software architecture proved scalable with an exercise in March of 1990 having some 850 objects at five sites, with most of those objects being semi-automated forces.
  - Objects in that test averaged one packet per second, with each packet being some 156 bytes in size for a peak requirement of 1.06 Mbits/second, just under the T-1 speed of the connecting links.

Distributed Interactive Simulation (DIS)

- The DIS network software architecture is a direct descendent from SIMNET but has packets that are more general than SIMNET's.
- DIS was an attempt to formally generalize and extend the SIMNET protocol.
- First version of the IEEE standard for DIS was appeared in 1993.
- Goals was to allow any type of player, on any type of machine, and to achieve large simulations.

Distributed Interactive Simulation (DIS)

- DIS has the same three basic components
  - an object-event architecture
  - notion of fully distributed simulation nodes
  - embedded set of predictive modeling algorithms for "dead reckoning"

Distributed Interactive Simulation (DIS)

- Protocol Data Unit (PDU)
  - The core of the DIS network software architecture is the protocol data unit (PDU).
  - Determining when each vehicle (node) of the simulation should issue a PDU is the key to this architecture.
  - The DIS (IEEE 1278) standard defines 27 different PDUs, only four of which (Entity State, Fire, Detonation, and Collision) are used by nodes to interact with the virtual environment.
  - In fact, most DIS-compliant simulations only implement those four PDUs, either throwing away the other 23 PDUs without comment or issuing a brief error message indicating a non-supported PDU was received.
  - A demonstration at the 1993 Interservice/Industry Training and Education Conference (I/ITSEC) showed that Entity State PDUs comprised 96% of the total DIS traffic.
  - Remaining 4% distributed mainly amongst Transmitter (50%), Emission (39%), Fire (4%), and Detonation (4%).

Distributed Interactive Simulation (DIS)

- Protocol Data Unit (PDU)
  - The simulation contained 79 players sending PDUs, though the actual mix of vehicles involved in this exercise is not available.
  - Air vehicles issued one ESPDU/second average in that demonstration, with land vehicles averaging 0.17 ESPDUs/second. Some participants in that demonstration issued packets at frame rate, and some produced 20 ESPDUs/second
  - In DIS, we get more of a notion that any type of computer plugged into the network that reads/writes DIS PDUs and manages the state of those PDUs properly can fully participate in a DIS environment.
  - This fully distributed, heterogeneous network software architecture means that workstation class machines can play against PC class machines.
  - Additionally, it means that the environment can include virtual players (driven by a live human at a computer console of some sort), constructive players (computer-driven players), and live players (actual weapons systems plugged into the DIS network).

DIS Scalability

- DIS Scalability
  - There are several instances of fairly large DIS engagements, much larger than the 300 to 500 players for which DIS is designed.
  - However, these "DIS" engagements actually modify the DIS network software architecture for their particular circumstances to achieve useful demonstrations.

A DIS Networked VE - CCTT

- The US Army's Close Combat Tactical Trainer (CCTT) is one of the larger scale networked virtual environments.

High-Level Architecture (HLA)

- Aims at providing a general architecture and services for distributed data exchange.
- While the DIS protocol is closely linked with the properties military units and vehicles, HLA does not prescribe any specific implementation or technology.
  - Could be used also with non-military applications (e.g. computer games).
  - Targeted towards new simulation developments
- HLA was issued as IEEE Standard 1516 in 2000.

Academic
Networked Virtual Environments

NPSNET

- The NPSNET Research Group is the longest continuing academic research effort in networked virtual environments. The focus of the group is on the complete breadth of human-computer interaction and software technology for implementing large-scale virtual environments (LSVEs).
- There have been several generations of software formally named NPSNET and several precursor systems.

Early NPS Networked VEs

- The origins of the NPSNET virtual environment are an introductory computer graphics class project, fall quarter 1986. In that class, two students, Doug Smith and Dale Streyle, developed a visual simulator for the fiber-optically guided missile (FOG-M) system [Smith/Streyle 87 and Zyda 88].
  - The implementation of FOG-M came at the same time that Mike Zyda went to Japan for an extended three week consulting visit.

Early NPS Networked VEs

- VEH was the vehicle simulator. VEH and FOG-M connected via a simple open socket that allowed VEH and FOG-M to do basic Unix read()/write() functions for exchange of state information. By July of 1987, the NPSNET group had networked FOG-M and VEH.

Early NPS Networked VEs

- The Moving Platform Simulator (MPS) was the NPSNET group's testbed for looking at how to achieve more players in the networked virtual environment. MPS-1, MPS-2 and MPS-3 utilized an ASCII, NPS-invented protocol and broadcasting to exchange state information (1988-1990)

NPSNET-1, 2 & 3

- NPSNET-1 was demonstrated live at the SIGGRAPH 91 conference as part of the Tomorrows Realities Gallery. NPSNET-1 did not use dead-reckoning. NPSNET-1 flooded the network with packets at frame rate.
- NPSNET-2 and 3 were utilized to explore better, faster ways to do graphics, and to extend the size of the terrain databases possible.
- NPS-Stealth was spawned off from NPSNET-1, with the goal of developing a system capable of reading SIMNET terrain databases and SIMNET networking protocols.
- NPS-Stealth was operational in March of 1993. It was the only workstation-based virtual environment capable of interoperating with the $350,000 per copy SIMNET system.

NPSNET-IV

- In March of 1993, Silicon Graphics came out with their Performer API for developing virtual environments and visual simulation systems. The NPSNET group looked at the SGI Performer demos and decided to build NPSNET-IV. NPSNET-IV was DIS-compliant, dead-reckoned and had spatial sound.
- NPSNET-IV has many capabilities that in some ways, make it one of the most ambitious virtual environments of its day. In Swiss-Army-knife-fashion, a player using NPSNET-IV can be a fully-articulated human, a ground vehicle of almost any type, an air vehicle of any type, and any type of surface and subsurface vessel.
- NPSNET-IV has interoperated with almost every DIS-compliant virtual environment ever constructed.

NPSNET

- NPSNET has gone through a number of versions ...

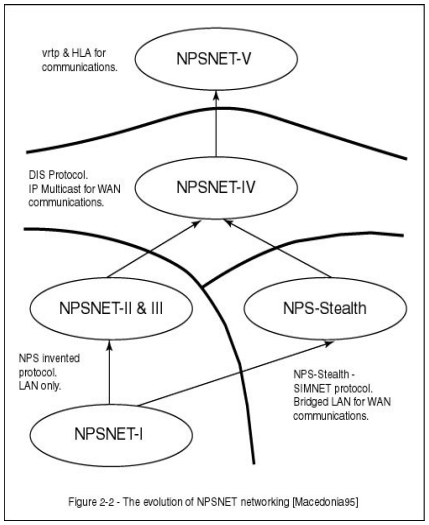


Figure 2-2 - The evolution of NPSNET networking [Macedonia95]

NPSNET-IV

- NPSNET-IV Capabilities
  - Building walkthroughs.
  - Articulated humans - mounting/dismounting capability.
  - Networking - play across the multicast backbone of Internet.
  - Terrain database integration, terrain paging (70km x 70km).
  - Any vehicle capability - air, ground, articulated human.
  - Testbed for VE NSA issues.
  - Interoperability - SIMNET/DIS
    - Constructive model integration - Janus World Modeler
    - ModSAF

NPSNET-IV

NPSNET-IV

NPSNET-IV

Paradise

- The PARADISE (Performance Architecture for Advanced Distributed Interactive Simulation Environments) project was initiated in 1993 by David Cheriton, Sandeep Singhal, and Hugh Holbrook of the Distributed Systems Group at Stanford University.
- PARADISE's designers focused on bandwidth reduction.
- The PARADISE system used IP multicast, assigning a different multicast address to each active object.
  - They did this by creating a mini-MBone among the non-multicast-capable hosts on an otherwise multicast-aware network.

Paradise

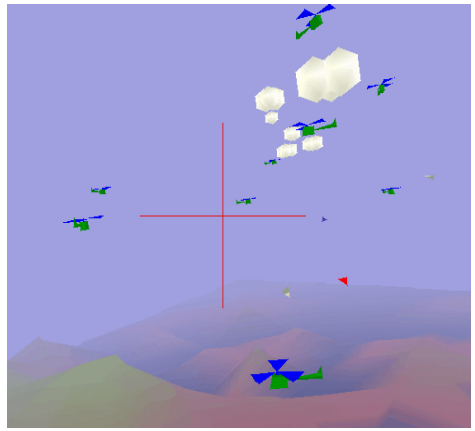
- Hosts transmit updates for local objects in much the same way as SIMNET and DIS.
  - However, to further reduce bandwidth, a hierarchy of Area of Interest (AOI) servers collect information subscriptions from each host.
  - The servers monitor the positions of objects and notify hosts about which objects' multicast groups they should subscribe to.
- Unlike SIMNET, PARADISE treats all objects, including terrain, uniformly as first-class entities.
  - Each is capable of transmitting state updates.
  - At the same time, PARADISE's designers tried to correct several of the mistakes made by DIS. For example, PARADISE recognizes that entities represented a spectrum ranging from rapidly-changing objects that needed to generate frequent updates to slowly-changing objects that rarely needed to send updates.

Paradise

- PARADISE supports multiple independent communication flows per object, with each flow enabling remote dead reckoning at a different level of accuracy.
- PARADISE also provides techniques for combining information about groups of objects, based both on their virtual world location and based on their type.

Paradise Scalability

- The system's flying helicopter demonstration could support 50-70 simultaneous entities, being bound by performance of the graphics hardware.



DIVE

- The Swedish Institute of Computer Science's Distributed Interactive Virtual Environment (DIVE) is another early academic virtual environment.
- DIVE has a homogeneous distributed database like SIMNET and DIS-compliant systems.
- DIVE uses the ISIS toolkit to implement the concept of process groups.
- However, unlike SIMNET the entire database is dynamic and uses reliable multicast protocols to actively replicate new objects.

DIVE

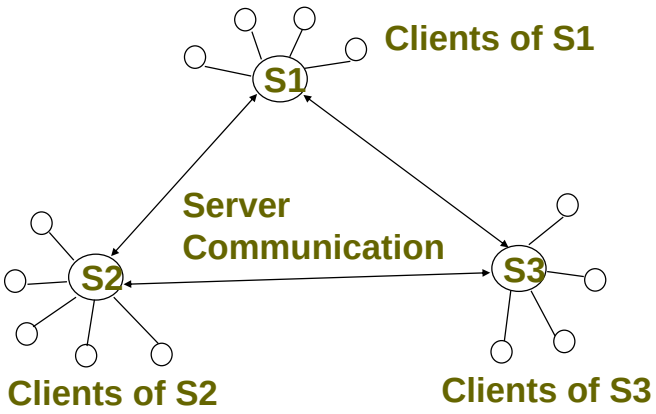
DIVE

- A disadvantage with this approach is that it is difficult to scale-up because of the communications costs associated with maintaining reliability and consistent data.
- For example, modeling terrain interactions, such as building a berm, still would be very expensive (though highly desirable) in terms of the number of polygons that would need to be created, changed, and communicated in DIVE.

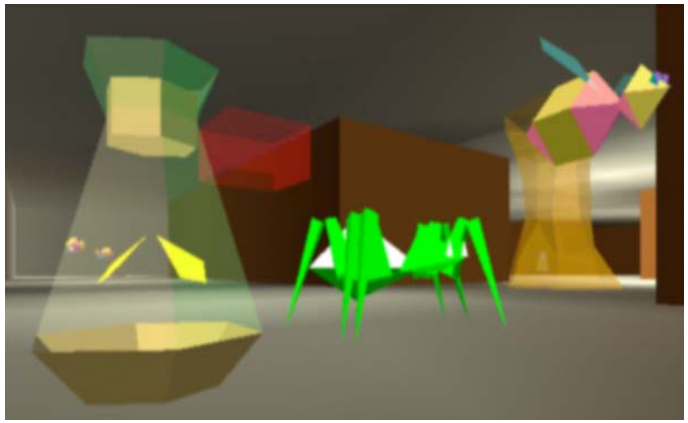
BrickNet

- BrickNet is the work of Gurminder Singh at the Institute of Systems Science at the National University of Singapore.
- A client-server model in which the database is partitioned among clients.
- Communication is mediated by central servers.
  - For example, as an entity moves through the VE, its database is updated by an object-request broker on a server that has knowledge of which client maintains that part of the world.
  - BrickNet shows us the limitations/possibilities of client-server architectures for Net-VEs.

BrickNet



BrickNet

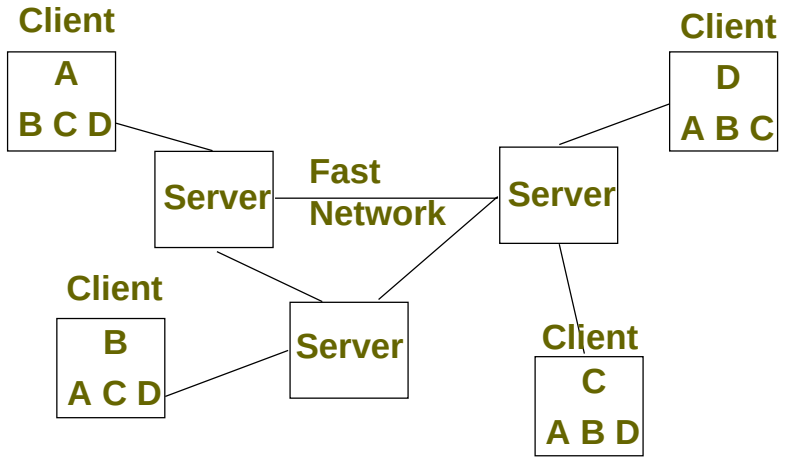


CyberBug Demo

RING: A Client-Server System for Multi-User VEs

- A client-server system similar to BrickNet that supports real-time visual interaction between a large number of users in a shared 3D VE.
- The key feature of the system is that server-based visibility algorithms compute potential visual interactions between entities representing users in order to reduce the number of messages required to maintain consistent state among many workstations distributed across a wide-area network.
  - When an entity changes state, update messages are sent only to workstations with entities that can potentially perceive the change - i.e. ones to which the update is visible.
  - Experiments in the paper show a 40x decrease in message traffic processed by client workstations during tests with 1,024 entities interacting in a large densely occluded VE.

RING: A Client-Server System for Multi-User VEs



RING: A Client-Server System for Multi-User VEs

- Every RING entity is managed by exactly one client workstation.
- Clients execute the programs necessary to generate behavior for their entities.
- In addition to managing their own entities, clients maintain surrogates for some entities managed by other clients (remote entities - like dead reckoning "ghosts" of the SIMNET and DIS standard).
- Communication between clients is managed by servers.
  - Again, this is the ideal organization for revenue generating networked games and it is not surprising to see a phone company developing such ideas.

RING: A Client-Server System for Multi-User VEs

- The primary advantage of the RING system design is that the storage, processing, and network bandwidth requirements of the clients workstations are not dependent on the number of entities in the LSVE.
- Client workstations must store, simulate, and process update messages only for the subset of entities visible to one of the client's local entities.
  - So this can be a low-cost $500 home box...
- Another advantage of the architecture is that high-level management of the VE can be performed by servers without the involvement of every client.
  - For instance, adding/removing an entity to/from the VE requires the notification of only one server.
  - That server handles notification of other servers & clients.

RING: A Client-Server System for Multi-User VEs

- Disadvantages of RING: extra latency is introduced when messages are routed through servers.
  - Sometimes messages are routed through multiple servers.
  - Also computation at the server is a bottleneck.
- Tests performed - 16 clients with varying numbers of servers (1 to 16) with 256 entities distributed across all clients and servers --> real-time.

The MERL Implementation - Diamond Park

- The MERL Diamond Park VE is built using SPLINE (Scalable PLatform for INteractive Environments) which provides the implementation of locales & beacons.
- Diamond Park has multiple users that interact in the park by riding around on bicycles and talking to each other (Social VR).

MERL Efforts in Large-Scale Multi-User VEs

- Locales are an efficient method for managing the flow of data between large numbers of users in a LSVE.
- The concept of locales is based on the idea that while a VE may be very large, most of what can be observed by a single user at a given moment is local in nature.
- Locales divide a VE into compact regions that can be process separately. Here's how:
  - Separate multicast addresses - each locale is associated with a separate communication channel. This makes it possible for a process to attend to what is happening in some locales without expending any resources on the locales that are being ignored.

MERL Efforts in Large-Scale Multi-User VEs

- Separate coordinate systems - each locale has its own coordinate system. This gives the effect of always having high precision in whatever locale is the current focus of attention, with gradually decreasing precision for locales that are distant from the current focus of attention.
- Arbitrary geometry - rather than cutting up a VE by some regular pattern, locales are a basis for constructing a VE from pieces. The shape, size and relative orientation of individual pieces can be chosen freely based on whatever is most convenient from the perspective of designing the individual pieces themselves. This facilitates the combination of many pieces designed by many people into an LSVE.

MERL Efforts in Large-Scale Multi-User VEs

- Beacons - are a special class of objects that can be located without knowing what locale they are in (to solve the "how do I join the VE problem").
- Beacons act as a content-addressable index from tags to the multicast address of locales. They make it possible to decide what locales to attend to based on what the locales contain.
- Beacons broadcast messages about themselves via the multicast address of the locale they are in AND they broadcast messages about themselves via a special beacon multicast address.
  - To ensure that this mechanism is scalable to large VEs, a large number of potential beacon multicast addresses are provided.

MR Toolkit Peers Package

- Another early effort in networked virtual environments is the MR Toolkit Peer Package (MR-TPP), an extension of the University of Alberta's MR (minimal reality) Toolkit [Shaw1993].
- MR-TPP is based on User Datagram Protocol (UDP) packets for network communication.
- MR-TPP has a software architecture in which local copies of shared data are maintained in a distributed fashion.
- MR-TPP maintains a complete graph connection topology, which means that each MR process that wishes to communicate with other processes must open a connection.
- In the MR-TPP architecture, there is apparently no notion of predictive modeling similar to DIS' dead-reckoning.

CAVERNsoft/QUANTA

- C++ toolkit for building Tele-Immersive applications with special emphasis on networking
- Client-server topology
- Higher-level networking and database APIs & Tools for application developer modules
- Available for Windows, SGI IRIX, Linux, FreeBSD, Sun Solaris, HP Unix, WinCE
- Graphics support for IRIS Performer

Low-Level Components

- Most of these capabilities have demo programs
- TCP, UDP, multicast, HTTP
- UDP reflector and multicast bridge
- TCP reflector
- Remote procedure calling (RPC)
- Remote File I/O
- Client/Server Databases
- Parallel Socket TCP
- Reliable Blast UDP (RBUDP)
- Cross-platform Data Conversions
- Mutual exclusion and threading
- Performance Monitoring- Netlogger compatible
- Implemented across SGI, Windows9x/NT/2000, Linux, FreeBSD

High-Level Developer Modules

- Audio streaming
- Base and Articulated avatars
- VR navigation and collision detection
- VR picking and moving
- VR network dynamic coordinate system
- VR menus
- Speech recognition with IBM ViaVoice
- Collaborative Animator
- Collaborative application shell to jumpstart development

Initialization & Release

- QUANTAinit()
- QUANTAexit()

- Call these before program initialization and after the main loop has terminated
- Note that creating your own classes which use QUANTA will need to keep in mind the timing of QUANTAinit() calls.

TCP Socket Modules

- Reliable transmission of data over socket connections

- QUANTAnet_tcpClient_c
- QUANTAnet_tcpServer_c

Sample TCP Client Code

- See demos/network/tcp

```
client->connectToServer( ipAddr 20000 );

status = client->write(
        sendBuffer,   // data to transmit
        &dataSize,    // size of the data
        QUANTAnet_tcpClient_c::BLOCKING );

status = client->read(
        receiveBuffer,        // buffer for reading
        &dataSize,            // size of the data
        QUANTAnet_tcpClient_c::BLOCKING );
```

Sample TCP Server Code

- Maintain a list of all clients
- Check each client for data transmissions

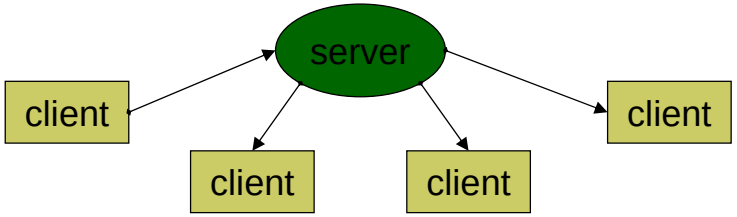
```
aClient = server->checkForNewConnections();
if (aClient)
        // register a new client

for (every client)
        status =  client->read(
                receiveBuffer[clientNumber],
                &dataSize,
                QUANTAnet_tcpClient_c::BLOCKING );
```

UDP Socket Modules

- Unreliable transmission of data over socket connection
- See demos/network/udp

- QUANTAnet_udpClient_c
- QUANTAnet_udpServer_c

UDP and TCP Data Reflection



- Clients send packets to the server, which in turn reflects those packets automatically to all other connected clients
- TCP reflector imposes "packet" boundaries
- QUANTAnet_tcpReflector_c
- QUANTAnet_udpReflector_c

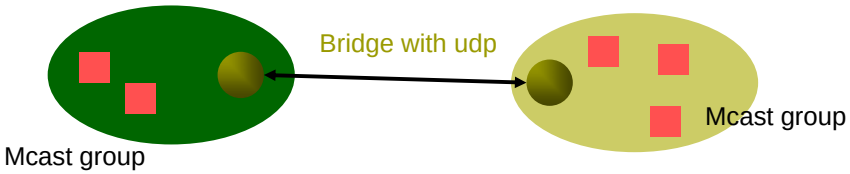
Sample TCP Reflector Code

```
server = new QUANTAnet_tcpReflector_c;
server->init();
server->interceptNewConnection(
        newclientcallback, NULL );

while(1) {
        server->checkForNewClients();
        server->process();
}
```

Unreliable Multicasting

- Allows for direct control over multicast connections
- QUANTAnet_udpReflector_c provides bridging between 2 multicast domains

- QUANTAnet_mcast_c



Parallel TCP Connections

- Multiple sockets are used concurrently to increase the data transfer rate
- Solve Long Fat Network problem without having to set TCP Window Size to Bandwidth Delay Product

- QUANTAnet_parallelTcpServer_c
- QUANTAnet_parallelTcpClient_c

```
client->connectToServer(
        iRemoteName,        // remote server
        iRemotePort,// remote port
        iPSocketSize );     // number of sockets
```

Reliable Blast UDP (RBUDP)

- RBUDP - An old idea that may be useful now that networking bandwidth is increasing
- Use UDP for bulk data transmission rather than TCP
- If bandwidth can be guaranteed by QoS –reliability will be high- chances of errors will be few

- QUANTAnet_rbudpReceiver_c
- QUANTAnet_rbudpSender_c

Datapacking

- Cross-platform data-packing class, for 32, n32 and 64 bit systems
- Takes care of Little Endian and Big Endian conversions
- Now has new array data packing methods

- QUANTAnet_datapack_c

Database / Persistence Distributed Shared Memory

- Data transmission is managed by path and key names
  - Paths group similar types of data
  - Keys are the individual packets of data
- Cross-platform data-packing class
- Provides persistent data!
- Persistence allows asynchronous retrieval of the state of the environment.
- Load from the database at program startup, save to the database at program exit

- QUANTAdb_client_c
- QUANTAdb_server_c

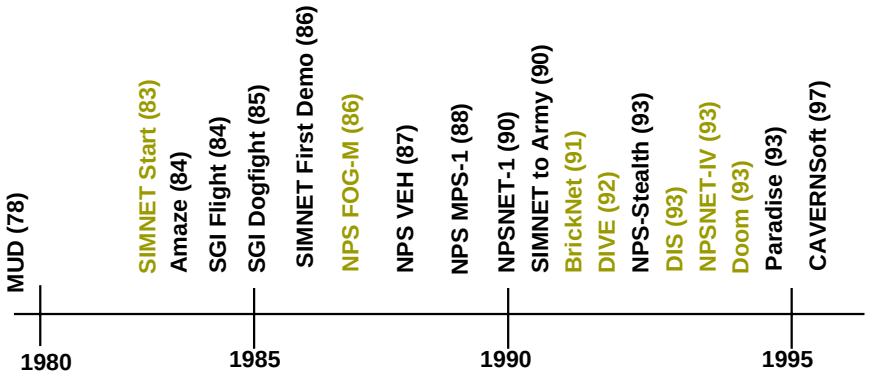
Shared State

- Encapsulates cross-platform network transmission of state information
- Intended to allow for transparent sharing

```
state.packAndSendState(); //packs the object state
and transmits it to the database server

state.unpackState( char *buffer ); // unpacks the object
state from the data buffer
```

- QUANTAdb_sharedState_c

A Brief Timeline of Networked-VEs



References

- Carlsson C. & Hagsand, O. DIVE - A platform for multi-user virtual environments, Computers & Graphics, 17(6), 663-669, 1993.
- Gossweiler, R. et al., An Introductory Tutorial for Developing Multiuser Virtual Environments, Presence 3(4), 1994, pp. 255-264.
- Funkhouser, Thomas A. "RING: A Client-Server System for Multi-user Virtual Environments," Proceedings of the 1995 Symposium on Interactive 3D Graphics, Monterey, April 1995, pp. 85-92.
- Funkhouser, Thomas A. "Network Topologies for Scalable Multi-User Virtual Environments," Proceedings of the 1996 VRAIS Conference, April 1996, pp. 222- 228.
- Barrus, John, Waters, Richard and Anderson, David B. "Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments," Proceedings of the 1996 VRAIS Conference, April 1996, pp.204-213.
- Leigh, J., Johnson, A., Brown, M., Sandin, D., DeFanti, T., Visualization in Teleimmersive Environments. In IEEE Computer, December, 1999, pp. 66-73