# Issues in the Design of a Flexible Distributed Architecture for Supporting Persistence and Interoperability in Collaborative Virtual Environments

**Jason Leigh**
**Electronic Visualization Laboratory,**
**University of Illinois at Chicago**
**spiff@eecs.uic.edu**
**http://www.evl.uic.edu/spiff**

**Andrew E. Johnson**
**Electronic Visualization Laboratory,**
**University of Illinois at Chicago**
**ajohnson@eecs.uic.edu**
**http://www.evl.uic.edu/aej**

**Thomas A. DeFanti**
**Electronic Visualization Laboratory,**
**University of Illinois at Chicago**
**tom@eecs.uic.edu**
**http://www.eecs.uic.edu/~tom**

## Abstract:

CAVERN, the CAVE Research Network, is an alliance of industrial and research institutions equipped with CAVE-based virtual reality hardware and high-performance computing resources, interconnected by high-speed networks, to support collaboration in design, education, engineering, and scientific visualization.

CAVERNsoft is the collaborative software backbone for CAVERN. CAVERNsoft uses distributed data stores to manage the wide range of data volumes (from a few bytes to several terabytes) that are typically needed for sustaining collaborative virtual environments. Multiple networking interfaces support customizable, latency, data consistency, and scalability that are needed to support a broad spectrum of networking requirements. These diverse database and networking requirements are characteristics typically unexhibited by previous desktop multimedia systems but are common in real-time immersive virtual reality applications.

## Keywords:

collaborative persistence virtual reality scalable

# 1 Introduction

Collaborative Virtual Reality (CVR) is currently one of the most challenging areas of research in Virtual Reality (VR) because it adds new dimensions to human-factors, networking, and database issues. For example,

human-factors research in VR has traditionally focused on the development of natural interfaces for manipulating virtual objects and traversing virtual landscapes. *Collaborative* manipulation, on the other hand, requires the consideration of how participants should interact with each other in a shared space, in addition to how co-manipulated objects should behave. Other issues include: how participants should be represented in the collaborative environment; how to effectively transmit non-verbal cues that real-world collaborators so casually and effectively use; how to best transmit video and audio via a channel that allows both public addressing as well as private conversations to occur; and how to sustain a virtual environment even when all its participants have left.

Naturally CVR poses new challenges to traditional areas of networking and databases as well. A Collaborative Virtual Environment (CVE) requires an unconventionally broad range of networking, database and graphics capabilities. This vast range makes the rapid construction of complex CVEs difficult. Current attempts at building networking and database architectures for CVEs have resulted in ad-hoc solutions that are specifically designed to solve a small range of problems[14, 2, 3, 20, 15, 23].

CAVERNsoft is a software architecture that attempts to be cognizant of the diverse human-factors, networking and database issues that are involved in supporting CVR. Its main goal is to explore the problem of supporting persistent CVEs for collaborative, education, and scientific and engineering visualizations that involve supercomputers as well as massive data stores. But its principals are generalizable to other CVR application domains.

In this paper we will describe a number of scenarios that have helped motivate the design of CAVERNsoft by identifying a number of key issues that a software architecture for CVR must support. We will then describe our architecture for CAVERNsoft.

# 2 Representative Collaborative Virtual Reality Scenarios

The following scenarios describe representative CVEs in several domains. These CVEs involve tasks that can benefit from a solution in CVR over simply non-collaborative VR or 3D workstation computer graphics.

## 2.1 Design and Engineering

Collaborative design work in VR typically involves a small group of users, either synchronously or asynchronously, engaged in the construction, and manipulation of objects in the virtual world. Since the interfaces for three-dimensional modeling in VR are still relatively imprecise compared to 2.5D CAD packages, most of the collaborative tasks in collaborative design involve evaluations of the design, and to a lesser degree, redesign or brainstorming for new design possibilities[13, 11, 12].

The National Computational Science Alliance (NCSA) has been working with Caterpillar Belgium S.A., to develop a system allowing remotely located engineers to work together on vehicle design review and redesign [10]. Remote collaboration is necessary here because the final system will be used by Caterpillar engineers in the U.S. and Europe who must jointly design Caterpillar vehicles so that they meet customer demands and safety requirements for both markets. For example, European safety standards require the addition of a roading fender. Virtual co-presence allows one designer to manipulate the fender while another designer watches for its effect on visibility from within the virtual cab of the vehicle.

## 2.2 Training

The earliest CVR systems were military-based applications such as SIMNET and NPSNET [14]. SIMNET is a standard for distributed interactive simulations developed by DARPA to facilitate training without the expense of conducting real battlefield exercises. One of the hundreds of SIMNET participants may be a foot-soldier wearing a head-mounted display and standing on a tread-mill while another may be sitting in a tank simulator. SIMNET's underlying unit of data transmission specifically contains encodings for military entities. DIS is a newer and more ambitious simulation standard based on SIMNET but allowing for greater complexity and realism.

These military simulations represent one extreme of collaborative VR where the emphasis is on reducing

networking bandwidth, latency and jitter to allow hundreds of participants to exist in the environment simultaneously. This is a classic example of the real-time nature of collaborative VR that current desktop multimedia systems do not address. The usefulness of the system depends on it being able to obtain minimum stable thresholds of network latency.

### 2.3 Scientific Visualization

A typical scenario in collaborative scientific visualization involves a small group of remotely located scientists entering a CVE to discuss a visualized data set. This data set may originate from a database or may be computing simultaneously on a supercomputer, in which case the virtual environment can be used to steer the computation.

Argonne National Laboratory (ANL) in collaboration with Nalco Fuel Tech have built an immersive interactive engineering tool for designing pollution control systems for commercial boilers and incinerators [7]. Multiple CAVEs can synchronously connect with an IBM SP supercomputer to steer the interactive simulation of flue gas flow in the boiler.

## 2.4 CALVIN and NICE - the Pre-history of CAVERNsoft

CAVERNsoft is a continuation and generalization of the work we have done at the Electronic Visualization Laboratory on several collaborative virtual environments in recent years. Two of these environments: CALVIN and NICE, provided testbeds to prototype several of the ideas that would eventually form parts of CAVERNsoft.

### 2.4.1 CALVIN - Collaborative Architectural Layout Via Immersive Navigation

CALVIN [12] is a CVE that allows multiple users to synchronously and asynchronously experiment with architectural room layout designs in the CAVE (Figure 1.)

Participants are able to move, rotate, and scale architectural design pieces such as walls and furniture. These participants may work as either ``mortals'' who see the world life-sized, or as ``deities'' who see the world as if it were a miniature model.

Asynchronous access allows designers to enter the space whenever inspiration strikes them, rather than requiring them to wait to schedule formal meetings, which can be particularly difficult if the participants are located at opposite parts of the world with significant timezone differences. In fact CALVIN already provides interfaces for bilingual (Japanese and English) interaction.

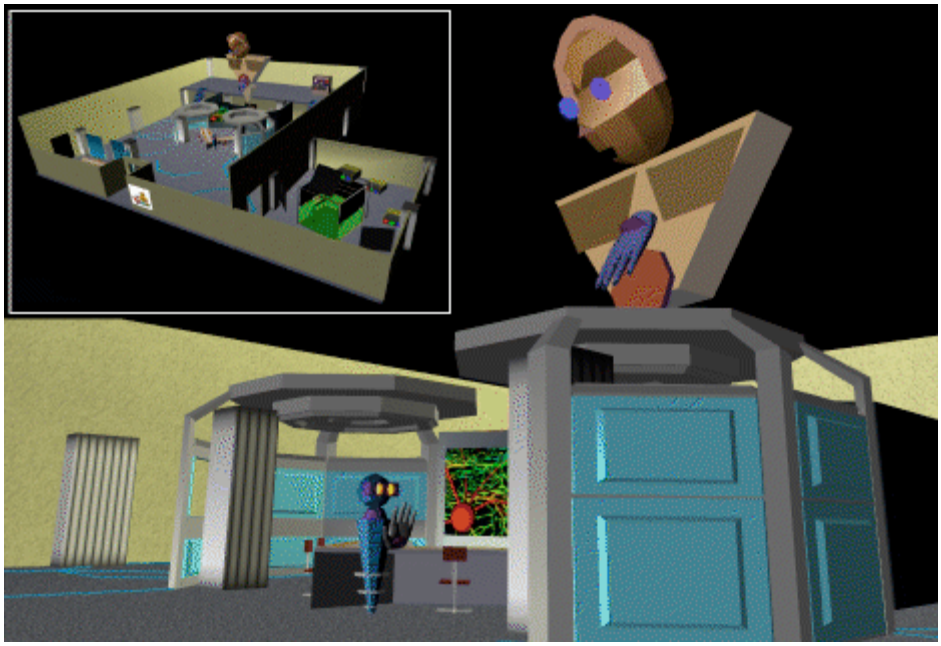


**Figure 1:** CALVIN: a collaborative design environment for architectural layout. The scene shows two avatars (a tall one and a small one) viewing the space at different perspectives. The top lefthand inset is a zoomed-out view of the entire design space.

CALVIN employees a shared variable model of a distributed shared memory (DSM) system to eliminate the need of the programmers to develop specific protocols for network communication. The DSM itself uses a reliable protocol and a centralized sequencer to guarantee consistency in all clients. C++ classes representing networked versions of floats, integers and character arrays are provided so that assignment to variable instantiations of these classes automatically shares the information with all the remote clients.

These networked variables are used to send data such as the state of objects in the world and user-tracker information. Tracker information is sent so that avatars can be drawn in the place of participants in the virtual scenes. Position as well as orientation data from the user's hand and head are transmitted so that fundamental gestures such as nodding, pointing, and waving can be communicated through the avatars.

Although the task of world synchronization is greatly simplified by the centralized sequencer, the transmission of tracker information over such a reliable channel can introduce latencies- especially when synchronizing between the participant's real location and their avatar's location. This is acceptable for small relatively closely located working groups where the network traffic and latency is relatively low but is unsuitable for larger and more distant groups of participants dispersed over the internet. In fact, to transmit audio/video signals between sites, the shared memory system is bypassed with point-to-point raw ATM streams which are able to support teleconferencing at NTSC resolution and at 30 frames per second.

Finally, in CALVIN when two or more participants simultaneously modify an object, a ``tug-of-war'' occurs where the object appears to jump back and forth between two positions, eventually remaining at the position given to it by the last person holding onto it. This problem can be alleviated by using a locking scheme, but this was intentionally **not** done. In VR, where emphasis is placed on natural interaction, it would be unnatural if the user had to lock an object before picking it up. The presence of avatars in combination with audio communication (the most important of the communication channels to provide) compensated for the lack of strict floor control and database locking. For example, the declaration: ``I'm going to move this chair'' combined with the visual cue of an avatar standing next to a chair and pointing at it, alerts other users that this user is about to grab that chair.

### 2.4.2 NICE - Narrative Immersive Constructionist/Collaborative Environments

The NICE group is building a collaborative environment in the form of a virtual island for young children (approximately 6-8 years of age)[19]. In the center of this island the children can tend a virtual garden. The children, represented by avatars, collaboratively plant, grow, and pick vegetables and flowers. They ensure that the plants have sufficient water, sunlight, and space to grow, and need to keep a look out for hungry animals which may sneak in and eat the plants. The children can shrink down to the size of a mouse and crawl under the garden to see the root system, and can talk with the other remotely located children or other characters in the scene. The children are able to modify the parameters of this small ecosystem to see how it affects the health of the garden (Figure 2.)

**Figure 2:** NICE: a narrative immersive collaborative environment for education. The top, left thumbnail shows an avatar handing a flower to another avatar in the NICE garden. The top right thumbnail shows a child interacting with an avatar in the CAVE. The bottom, left thumbnail is a snapshot of the NICE Java interface. The bottom, right thumbnail is a snapshot of NICE in a VRML2 browser. Click on the thumbnails to view an enlarged version of each image.

NICE's architecture is based on the techniques derived from CALVIN in that a central server is used to maintain consistency across all the participating virtual environments. Whereas CALVIN solely used a reliable connection to synchronize state information, NICE used an unreliable protocol (either multicasting or UDP) to share avatar information from magnetic trackers, and a reliable socket connection to share world state information and to dynamically download models from WWW servers using the HTTP 1.0 protocol.

Both multicasting and UDP were provided to deliver tracker data, as it was not always possible to acquire the administrative privileges to conveniently erect multicast tunnels between distant remote sites. Hence a number of interconnected NICE ``smart-repeaters'' were deployed at various remote sites that allowed the use of multicasting amongst clients at localized sites but UDP for repeating packets between remote locations. In addition, to prevent faster clients from overwhelming slower clients with data, the smart-repeaters performed dynamic filtering of data based on the throughput capabilities of the clients. Using this scheme participants running on high speed networks have been able to collaborate with participants running on slower 33Kbps modem lines.

NICE's virtual environment is persistent. That is, even when all the participants have left the environment and the virtual display devices have been switched off, the environment continues to evolve; the plants in the garden keep growing and the autonomous creatures that inhabit the island remain active.

Interactions with the NICE garden are not limited to users with VR hardware. The garden in NICE can be experienced either by entering VR, a basic WWW browser (http://www.ice.eecs.uic.edu/~nice), a VRML2 browser, or in a Java applet. Participants using a mouse can interact with participants using VR hardware where the desktop user's mouse position is used to position an avatar in the 3D virtual world, and the bodies of the VR users are used to position 2D icons on the desktop screen. This kind of scalability will be important for increasing the breadth of possible collaborations.

# 3 The Particular Requirements of Collaborative Virtual Reality

The above scenarios illustrate the broad spectrum of human-factors, graphics, networking, and database requirements that are needed to support CVR. These requirements are compiled in the following sections.

## 3.1 Avatars

The elaborateness of the avatar should vary with the task being performed. Hence it is important to identify the minimum elements of representation needed to afford recognizability and to convey non-verbal information such as body language and gesture. In our experience we have found a minimum of head position and orientation, body direction, and hand position and orientation to be adequate for many CVR tasks. To afford recognizability, we have found it easier to distinguish avatars based on geometry rather than color. Hence the commonly used, homogeneously shaped avatars with varying colors and overlayed name tags, do not make good avatars.

To support the minimal avatar, a bandwidth of approximately 12Kbits/sec (at 30 frames per second) is needed. Theoretically this implies that 10 avatars can be supported over a 128Kbits/sec ISDN connection. In practice however, our experiments have shown that it is able to support a maximum of four avatars with an average latency of 60ms using UDP as the transmission protocol. Although this is not a scalable solution, it is a cost effective means of transmitting VR avatar data with the quality of service of a dedicated connection.

## 3.2 Suitable Interfaces for Collaborative Manipulation and Visualization

High-level virtual interfaces must be developed to allow collaborative manipulation of shared objects. In addition, these manipulation tools require some form of locking to occur so that consistency is maintained across all the virtual environments sharing the virtual space. The goal is to provide mechanisms for acquiring distributed locks (possibly through predictive means) so that the user does not realize that locks have had to be acquired before objects could be manipulated. This is particularly important over high latency networks where there might be a noticeable delay between the time when a user physically picks up an object (and hence attempting a lock on it,) to the time when the VR system confirms the lock on the object. Lag similar to this has been shown to significantly degrade human performance in a VR environment[24]. In our experience we have found that for coordinated VR tasks involving two expert VR users, performance begins to degrade when network latency increases above 200ms[18]. Other research has found acceptable latencies to be much lower (100ms)[14]. The acceptable latency is expected to be lower for inexperienced users and for coordinated tasks involving very fine manipulation of shared objects. In such situations tracker inaccuracy will also begin to affect human performance.

## 3.3 Audio/Video Teleconferencing

Audio (voice telephony) is one of the most important channels to provide in a collaborative experience[5, 22]. It has been shown that latencies of greater than 200ms will result in degradations in conversion[4]. As the latencies continue to increase the amount of time spent in confirming conversion increases, and the amount of useful information being conveyed in the conversation decreases. Video conferencing is useful in instances where it is important for the participants to see each other face to face for negotiation tasks[1, 16, 21]. In traditional conference-room video conferencing, video provides a means to convey a sense of co-presence[17]. In VR however this sense is created through the use of avatars and hence we believe video will play a less

significant role in the collaboration.

## 3.4 Flexible Support of Various Data Characteristics

The design of the CVR library is dependent on two interrelated factors: the characteristics of the data being distributed and the distribution scheme employed. The four attributes that characterize CVR data that most greatly affect the mode of transmission, management and storage of CVR data, are: quality of service, data size, persistence and queueing.

### 3.4.1 Quality of Service

For closely coordinated work in CVR, minimum levels of network bandwidth, latency and jitter are desirable. In addition, both reliable and unreliable protocols of unicast, broadcast and multicast transmission are needed to optimally transport different classes of CVR data (3D tracker data, state information, streamed audio/video feeds, geometric models, large scientific data sets.)

### 3.4.2 Data Size

There are essentially three categories of CVR data sizes: small-event, medium-atomic, and large-segmented. These divisions are created because they affect the manner in which they are optimally transmitted.

- **Small-Event data** are data such as unreliable tracker data, and reliable state and event data. These typically require priority transmission with low latency.
- **Medium-Atomic data** are data that are small enough to fit in the physical memory of the client because it must be processed as one atomic ``chunk.'' Examples of these are 3D geometries representing individual objects in the VR scene.
- **Large-Segmented data** are data that are too large to fit in the physical memory of the client and hence can only be accessed in smaller segments. Large scientific data sets and long pre-digitized video streams fit this category. These data sets usually need to be "abstracted-down" first before they are visualized as the amount of data that can potentially be visualized can easily exceed the graphics rendering capabilities of the VR system.

### 3.4.3 Queued/Unqueued Data

Data that are sent to clients or servers, regardless of whether they are stored in a database or not, need to be either queued or unqueued. For example, world state information may be unqueued since only the latest information is necessary. Queued data are data which must all arrive at a client or server in order. This implies the use of a reliable protocol. There are however instances where a queued, unreliable protocol may still be useful- specifically for audio conferencing, long, unreliable data streams are transmitted to all participating clients.

### 3.4.4 Persistent/Transient Data

Persistent data characterizes data that needs to be stored in a database for later use. This data remains in the database after all the clients leave the CVE. All state data that is crucial to the resumption of a client in a CVR session must be persistent. Models and scientific datasets that will be loaded into CVE are also prime candidates for database storage.

Transient data are data that are not stored in a database. An example of this kind of data are command messages that might be sent between clients to effect events or audio/video data streams. An exception to this definition is when transient data is stored in a database to allow re-play of events at a later time. In this case the data is more accurately characterized as persistent rather than transient.

## 3.5 Scalable and Flexible Topological Construction

No single interconnection of distributed resources will perform optimally for all CVR applications. The number

of participants expected to work in the environment and the amount and form of the data being shared has profound effects on the design of the distributed topology. Systems that are designed to scale well with respect to connectivity (connection scalability) typically must sacrifice strong data consistency. Most currently existing systems prioritize connection scalability over data scalability (ability of CVEs to handle enormous amounts of data.)

It is our belief that data scalability is of greater importance to the development of engineering and scientific applications than connection scalability. Data sets in these problem domains are typically enormous in size however the number of people simultaneously collaborating is unlikely to exceed 6 or 7.

The three main classes of distributed topologies used in CVR include: replicated homogeneous, shared centralized, and shared distributed[14]. These are described below.

- **Replicated Homogeneous**

  Replicated Homogeneous topologies are classical of military VR simulations (as in SIMNET, NPSNET, DIS)[14]. In such topologies each client holds a completely replicated database of the shared environment and state information is shared by broadcasting messages to all participating clients. This system has no centralized control whatsoever, hence any new client joining a session must wait and gather state information about the world that is broadcasted by the other clients.
- **Shared Centralized**

  In this approach all shared data is stored at a central server. The main advantage of this scheme is that it greatly simplifies the management of multiple clients, especially in situations requiring strict concurrency control. However, its role as an intermediary for the delivery of data can impose an additional lag in the system. Another disadvantage is that if the central server fails none of the connected clients can interact with each other. Despite these disadvantages, this architecture is still useful for supporting small groups of collaborators.

- **Shared Distributed with Peer-to-peer Updates**

  This approach simulates a wide-area shared memory structure[3, 20, 15, 23] in which objects that are instantiated at one site are automatically replicated at all the remote sites. This logical abstraction simplifies the CVR application development at the cost of performance. Typically in these implementations, a newly connected client must form point-to-point connections with all the participating clients. Hence for $n$ participants the number of connections required is $n(n\text{-}1)/2$. In addition if the environment involves the sharing of enormous scientific data sets, the data set will be fully replicated at every site. Unless the data sharing policy is modified to account for large datasets this scheme will not be scalable.

- **Shared Distributed using Client-server Subgrouping**

  This topology distributes the database amongst multiple servers. Clients connect to the appropriate server as needed. A classic approach is to bind the servers to unique multicast addresses. Clients then subscribe to different multicast addresses to listen to broadcasts from the servers[2, 8].

## 3.6 Synchronous and Asynchronous Collaboration

The main focus of most CVR applications has been on synchronous collaboration. That is, all participants are working together in the environment at the same time. However in trans-global collaborations the timezone differences make routine synchronous collaboration highly inconvenient. In this case it is important to also provide a means for distributed groups to work asynchronously in a shared virtual space. The support of asynchrony will require the use of distributed databases to maintain the states between the remote sites.

## 3.7 Persistence in Collaborative Virtual Reality

Persistence in Collaborative Virtual Reality describes the extent to which the virtual environment exists after all participants have left the environment. Persistence can be divided into three major classes: participatory persistence, state persistence, and continuous persistence.

- **Participatory Persistence**

  This is persistence in which the VE only exists in the brief amount of time that participants are in it. When all participants leave, the environment is extinguished with no record of the state of the environment before it was extinguished. When the environment is started at a later time, it always begins at the beginning. Most virtual environments are still only participatory persistent.

- **State Persistence**

  This is where the state of the virtual environment may be saved at any given time to be recalled later. Either intermittent snapshots can be created or entire collaborative experiences can be recorded for later review. This form of persistence can be used to support version control and annotations made in CVR.

- **Continuous Persistence**

  This is where the state of the virtual environment remains extant even when all the participants have left. Hence when participants re-enter the environment the state of the world may have changed. Such environments are liken to MUDs (Multiuser Domains/Dungeons) which by their popularity, have shown to encourage the spontaneous use of collaborative environments.

### 3.8 Interoperability with Heterogeneous Systems

The varying problem domains in which CVR is applied requires connectivity between heterogeneous resources such as external databases, supercomputers, desktop workstations, and miscellaneous VR systems. For example, Argonne's incinerator simulator connects the CAVE VR system to an IBM SP supercomputer. The supercomputer performs the computation while the CAVE visualizes the results.

### 3.9 Application Specific Servers

These are unlike traditional networking and database servers in that they do not simply store and forward data. Application specific servers in VR also possess semi-graphical capabilities as they may need a local representation of the virtual space for their operation. For example, an application specific server simulating the movement of autonomous agents through a virtual landscape may also use the same graphical routines that model and visualize the terrain to perform operations such as collision detection.

## 4 CAVERN

CAVERN is our proposed solution for supporting the full spectrum of CVR requirements described above. CAVERN (CAVE Research Network) is a collection of participating industrial and research institutions equipped with CAVEs, ImmersaDesks, and high-performance computing resources all interconnected by high-speed networks for the purpose of supporting collaborative: engineering and design; education and training; and scientific visualization and computational steering, in virtual reality.

CAVERNsoft is the collaborative software backbone for CAVERN. CAVERNsoft employs distributed data stores to manage the wide range of data volumes that are typically needed for sustaining persistence in virtual environments. CAVERNsoft provides multiple networking interfaces with customizable, latency, data consistency, and scalability that are needed to support the broad spectrum of CVR networking requirements. Finally CAVERNsoft provides a set of higher-level modules to facilitate rapid construction of CVR applications.

### 4.1 The Information Request Broker

The Information Request Broker (IRB) is the nucleus of all CAVERN-based client and server applications. An IRB is an autonomous repository of persistent data driven by a database, and accessible by a variety of networking interfaces. The goal is to develop a hybrid system that combines the distributed shared memory model from CALVIN with distributed database technology and realtime networking technology under a unified interface to allow the construction of arbitrary CVR topologies.

A client application is built by using an IRB interface (IRBi) which, on invocation, will spawn the client's ``personal'' IRB. This IRB is used to cache data retrieved from other IRBs during the operation of the client. An application specific server is similarly built using the IRBi. Hence there is actually little differentiation between a client and a server. Using the IRBi a client can arbitrarily form a connection with any other client or server to access its resources. The IRBi will communicate the request to the client's personal IRB which will then communicate with the remote client's or server's IRB. It is the IRBs' responsibility to negotiate the networking and database services requested by the client/server applications. This form of flexibility will allow arbitrary CVR topologies to be constructed (Figure 3.)

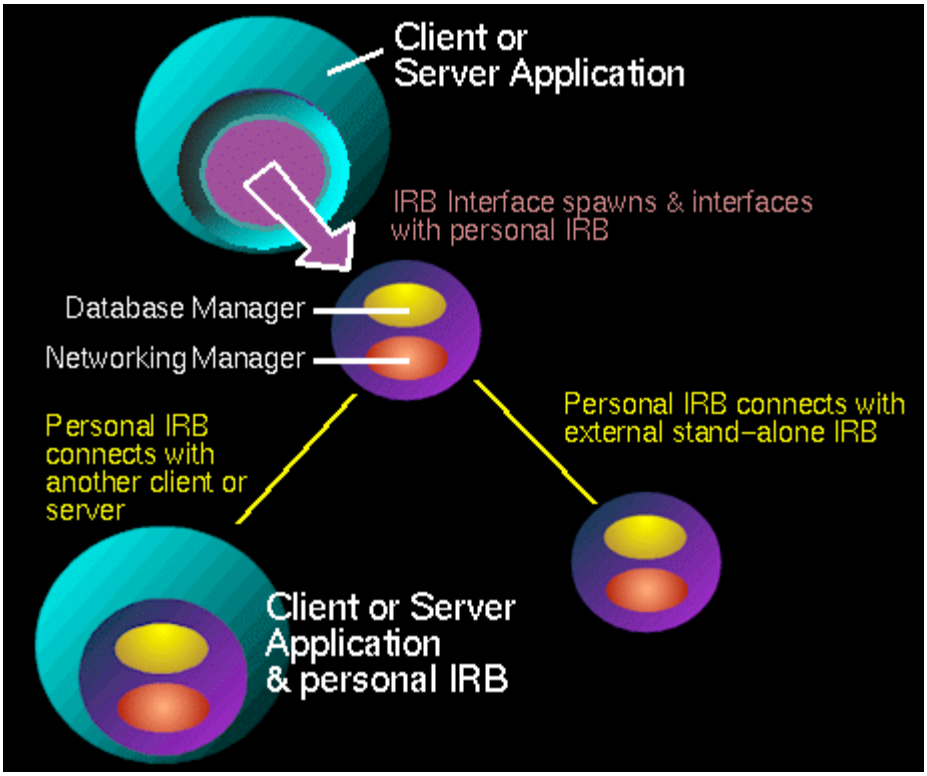


**Figure 3:** Clients/Servers use the IRB interface to spawn personal IRBs with which to communicate with other clients/servers or standalone IRBs.

## 4.2 The IRB Interface

The IRB interface (IRBi) is the client and server's interface to the IRB. The IRBi provides a networking interface, a database interface and a high-level template interface. The IRBi is tightly coupled with the IRB as they are merely threads that share the same address space. This reduces the need for creating artificial message passing schemes to invoke functionality between the client's application and the IRB.

A client's handle to their personal IRB is used to activate dynamic connections with remote IRBs. A client wishing to share information between its personal IRB and a remote IRB begins by first creating a communication channel and declaring its communication properties. Then any number of local and remote keys may be linked over the channel. A key is a handle to a storage location in an IRB's database. The database is used to cache data received from remote keys. Keys are uniquely identified across all IRBs and can be hierarchically organized much like a UNIX directory structure. Each local key may be linked to only one remote key. This is the link explicitly invoked by the user to share information with a remote IRB. However each local key can *accept* multiple linkages from other remote subscribing keys. The user is generally made unaware of these additional linkages as the personal IRB transparently manages data sharing with the remote subscribers. Any modifications that are made to one key will automatically be propagated to all the other linked keys.

### 4.2.1 Channel Properties

Channel properties allow clients to specify the networking service desired for data delivery. Clients may specify reliable TCP, or unreliable UDP and multicast. Large packets delivered over unreliable channels will

automatically be fragmented at the source and reconstructed at the destination. If any fragment is lost while in transit the entire packet is rejected.

In addition to connection reliability clients may specify Quality of Service (QoS) requirements. Hence they are able to declare the desired bandwidth, latency, and jitter of the data stream. The personal IRB will attempt to obtain the desired level of QoS from the remote IRB, but if it fails, the client may at any time negotiate for a lower QoS. As in RSVP[25] client-initiated QoS is used so that the client can specify the amount of data it can handle from the remote IRB.

### 4.2.2 Link Properties

Link properties allow clients to specify the actions taken when local and remote keys are linked. This includes being able to choose between active and passive updates and being able to select the initial and subsequent synchronization behavior.

In most CVR applications, world state information consisting of a few tens of bytes are actively distributed. That is, the moment a new value is generated it is automatically propagated to all the subscribers of the data. Passive updates occur only on subscriber request and usually involves a comparison of local and remote timestamps before transmission. For example, passive updates are typically used to download large volumes of 3D model data. Caching data and comparing their timestamps helps to reduce the need to redundantly download the same data set.

The initial synchronization behavior determines how the local and remote keys should be synchronized when the links are first formed. That is, clients are able to choose to synchronize automatically based on the keys' timestamps. That is the older key will be updated with information from the newer key. However the client may also choose to force synchronization from the local key to the remote key, and vice versa, regardless of timestamp. Of course clients may choose to perform no initial synchronization at all.

Subsequent synchronization behavior specifies the manner in which data is synchronized when local or remote updates to keys occur. The same options as for initial synchronization hold.

The default link property is to use active updates with automatic initial and subsequent synchronization.

### 4.2.3 Key Properties

Keys may be defined at a client's personal IRB or at a remote IRB provided the client has the necessary permissions. Keys may either be transient or persistent. Persistent keys are keys that will be stored in the IRB's datastore so that when a client or server re-launches, the data will still be retrievable by specifying the same key identifier. Clients determine whether a key is to persist by asking the IRB to perform a commit operation on the data. In addition simple locking functions are provided to allow clients to lock local or remote keys. Locking calls are non-blocking to prevent realtime applications from stalling when attempting to acquire locks on keys. Instead the locking call accepts a user-specified callback function that will be called when a lock has been acquired or when any relevant event pertaining to the lock occurs.

### 4.2.4 Asynchronous Triggering of Events

Many events may arise during the course of distributing data between clients and servers. The client/server may need to be notified so that appropriate actions may be taken in response to these events. It is inefficient for realtime VR applications to poll for such events. Instead the programs provide the IRBi with callback functions that the IRBi may call when the event arises.

Some examples of events include: new incoming data event; IRB connection broken event; QoS deviation event.

### 4.2.5 Recording Keys

In addition to declaring the retention properties of keys the IRBi allows the clients to declare keys that hold

recordings of groups of keys. This facility is provided to support State Persistence in VR.

In these recordings close synchronization of remote system clocks is not absolutely necessary as recording is always made from one point of view and hence it is the point of view's time reference that all relevant information is recorded.

Recordings may consist of time stamping and storing every change in value that occurs at a key and recording the state of all the keys at wide intervals. The former is needed to track the gradual changes in the virtual environment over time. The latter is needed to establish checkpoints so that the recordings may be fast-forwarded or rewound without having to compute every successive state that led to the fast-forwarded/rewound location.

On playback the recordings will populate the appropriate keys and, if desired, trigger client callbacks. In some instances it is useful to be able to playback only a subset of the recorded keys. This will allow the user to observe smaller subsets of events that occur in the VR environment.

Finally to synchronize the playback of experiences across multiple virtual environments each environment must constantly broadcast their frame-rate. This ensures that faster VR systems do not overtake slower systems while rendering the virtual imagery.

### 4.2.6 Direct Connection Interface

In addition to the many automatic networking capabilities provided by IRBs the IRBi must still support direct access to low-level socket TCP, UDP, multicast interfaces so that connectivity with legacy systems (such as WWW servers) can be supported. However CAVERNsoft adds value to the basic socket-level interfaces by providing automatic mechanisms for accepting new connections, and making asynchronous data-driven calls to user-defined callbacks.

### 4.2.7 Supplementary Concurrent Processing Facilities

Most of the networking and database operations performed in the IRB are executed concurrently and, if a multiprocessor system is available, in parallel with the VR system. It is therefore necessary to provide basic concurrency control primitives such as mutual exclusion and signals. These are implemented as macro definitions on top of the underlying threads library used by the IRB (for example POSIX threads.)

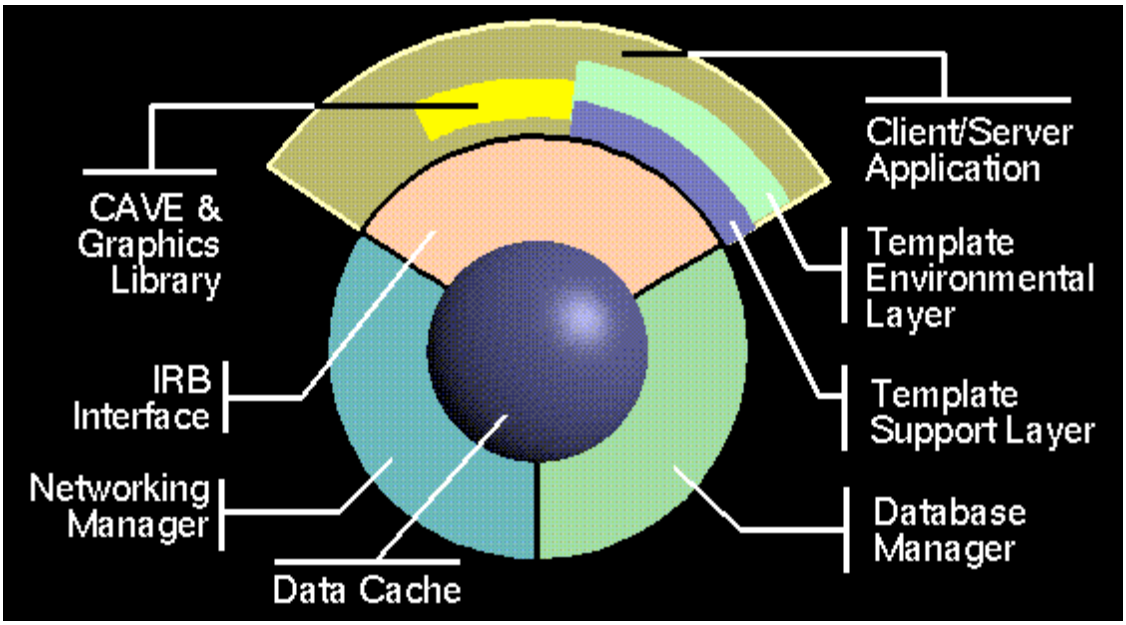


**Figure 4:** Software architecture of a CAVERN IRB-based Client/Server.

### 4.2.8 High-level Templates

In our experience with developing CVEs, we have found that it is substantially more difficult to retro-fit a single-user application with collaborative capabilities than to include these capabilities as part of the original design of the application. The IRB's high-level templates are designed to encourage VR application developers to think in terms of developing a collaborative application, by providing them with high-level tools that they are likely to all commonly need.

Templates are divided into two categories: support templates and environmental templates.

Support templates provide a collection of libraries to support various basic CVR component services such as: encoding and decoding of audio and video streams for teleconferencing and management of avatars.

Environmental templates provide a suite of complete but extensible CVEs. For example an environmental template could be designed specifically to help domain scientists ``jumpstart'' the process of building collaborative scientific visualization applications. Such a template would automatically provide networking, visualization and recording components as well as basic collaboration components such as audio/video conferencing, and avatars.

The template layers of CAVERNsoft are the only layers that simultaneously interface with the IRB interface and the graphics interface (Figure 4.) For example, in order to support avatars, the IRB interface must be used to declare keys to hold avatar information. The graphics portion of the avatar template will be implemented with the CAVE library in conjunction with either OpenGL, OpenInventor, or Performer. This separation of the basic IRB interface and the graphics interface with the template layer allows the IRB system to be used in non-graphic computing systems such as supercomputers or workstation farms.

### 4.3 Implementation Notes

Figure 4 shows how all the various components of CAVERNsoft fit together to build a client/server application. The networking manager is founded on Nexus[6]. Nexus is an efficient multithreaded communications library developed by Argonne National Laboratory to connect client applications with remote supercomputing resources. Using Nexus the IRB's networking manager can negotiate networking protocols and quality of service contracts, and manage connections once they have been established. As Nexus was originally built to coordinate communications amongst multiple nodes of supercomputers this allows CAVERNsoft to leverage those capabilities in support of IRB-based clients that will run on supercomputers.

The database manager will be built using PTool[9], a persistent object store developed by the Laboratory for Advanced Computing at the University of Illinois at Chicago. PTool's main use is in the efficient storage and retrieval of enormous persistent objects (typically occupying giga- to tera-bytes in size). A custom interface will be built on top of PTool to provide the abstraction of persistent keys. Strictly speaking the database that CAVERNsoft uses is a datastore. PTool achieves significant performance improvements over other object-oriented databases by stripping away the transaction management capabilities found in traditional databases.

## 5 Future Work

This paper has outlined the major issues in data distribution in persistent collaborative VR environments. CAVERNsoft's design goal has been to take into account a broader range of CVR requirements than any other CVR software infrastructure to date. We believe this broad-based approach will result in a highly flexible CVR software backbone that will enable the construction of a new generation of CVR applications.

We are now in the process of implementing a prototype of the IRB. In addition we are continuing to perform human-factors experiments to better understand the effect of network latency and jitter on coordination between remote users in CVR. This will be useful to further prescribe the performance requirements of CAVERNsoft.

## Acknowledgements

Major funding is provided by the National Science Foundation, CDA-9303433.

# References


**1**

ARGYLE, M., AND COOK, M. *Gaze and Mutual Gaze*. Cambridge University Press, 1976.

**2**

BARRUS, J. W., WATERS, R. C., AND ANDERSON, D. B. Locales and beacons: Efficient and precise support for large multi-user virtual environments. In *Proceedings of the Virtual Reality Annual International Symposium. VRAIS'96* (March 1996), IEEE Computer Society, IEEE, pp. 204-213.

**3**

CARLSSON, C., AND HAGSAND, O. DIVE - a multi-user virtual reality system. In *Proceedings of the IEEE Virtual Reality Annual International Symposium* (1993).

**4**

FISH, R. Bellcore cross-industry working team workshop XIWT. personal correspondence, 1996.

**5**

FISH, R. S., KRAUT, R. E., AND CHALFONTE, B. L. The videowindowsystem in informal communication. In *Proceedings of Computer Supported Cooperative Work* (1990), pp. 1-11.

**6**

FOSTER, I., KESSELMAN, C., AND TUECKE, S. The Nexus approach to integrating multithreading and communication. *Journal of Parallel and Distributed Computing*, 37 (1996), 70-82.

**7**

FREITAG, L., DIACHIN, D., HEATH, D., HERZOG, J., AND PLASSMANN, P. Remote engineering using cave-to-cave communications. *Virtual Environments and Distributed Computing at Supercomputing'95: GII Testbed and HPC Challenge Applications on the I-Way* (1995), 41.

**8**

FUNKHOUSER, T. A. Network topologies for scalable multi-user virtual environments. In *Proceedings of the Virtual Reality Annual International Symposium. VRAIS'96* (March 1996), IEEE Computer Society, IEEE, pp. 222-228.

**9**

GROSSMAN, R. L., HANLEY, D., AND QIN, X. PTool: A light weight persistent object manager. In *Proceedings of SIGMOD'95* (1995), ACM, p. 488.

**10**

LEHNER, V. D., AND DEFANTI, T. A. Distributed virtual reality: Supporting remote collaboration in vehicle design. *IEEE Computer Graphics and Applications, in press* (1997).

**11**

LEIGH, J., JOHNSON, A., AND DEFANTI, T. A. CALVIN: an immersimedia design environment utilizing heterogeneous perspectives. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems '96* (June 1996), pp. 20-23.

**12**

LEIGH, J., AND JOHNSON, A. E. Supporting transcontinental collaborative work in persistent virtual environments. *IEEE Computer Graphics and Applications* (1996), 47-51.

**13**

LEIGH, J., JOHNSON, A. E., VASILAKIS, C. A., AND DEFANTI, T. A. Multi-perspective collaborative design in persistent networked virtual environments. In *Proceedings of IEEE Virtual Reality Annual International Symposium '96* (Apr. 1996), pp. 253-260.

**14**

MACEDONIA, M. R., AND ZYDA, M. J. A taxonomy for networked virtual environments. In *Proceedings of the 1995 Workshop on Networked Realities* (Oct 1995).

**15**

MANDEVILLE, J., FURNESS, J., AND KAWAHATA, T. Greenspace: Creating a distributed virtual environment for global applications. In *Proceedings of IEEE Networked Virtual Reality Workshop* (1995), IEEE.

**16**

MCGRATH, J. *Groups: Interaction and Performance*. Englewood Cliffs, NJL Prentice-Hall, 1984.

**17**

OLSON, J., AND OLSON, G. M. What mix of video and audio is useful for small groups doing remote real-time design work. In *Proceedings of SIGCHI'95* (1995), ACM, ACM Press, pp. 362-368.

**18**

PARK, K. S. Effects of network characteristics and information sharing on human performance in cove. Master's thesis, Electronic Visualization Laboratory, University of Illinois at Chicago, 1997.

**19**

ROUSSOS, M., JOHNSON, A., LEIGH, J., BARNES, C. R., VASILAKIS, C. A., AND MOHER, T. G. The nice project: Narrative, immersive, constructionist/collaborative environments for learning in virtual reality. In *ED-MEDIA/ED-TELECOM 97: World Conferences on Educational Multimedia and Hypermedia and on Educational Telecommunications* (1997).

**20**

SHAW, C., AND GREEN, M. The MR toolkit peers package and environment. In *Proceedings of the Virtual Reality Annual International Symposium. VRAIS'93* (1993), IEEE Computer.

**21**

SHORT, J., WILLIAMS, E., AND CHRISTIE, B. *The Social Psychology of Telecommunications*. Wiley and Sons, 1976.

**22**

TANG, J. C., AND ISAACS, E. Why do users like video? In *Computer Supported Cooperative Work* (1993), CSCW, pp. 163-196.

**23**

WANG, Q., GREEN, M., AND SHAW, C. EM - an environment manager for building networked virtual environments. In *Proceedings of the Virtual Reality Annual International Symposium. VRAIS'95* (1995), IEEE Computer, IEEE, pp. 11-18.

**24**

WARE, C., AND BALAKRISHNAN, R. Reaching for objects in VR displays: Lag and frame rate. *ACM Transactions on Computer-Human Interaction 1*, 4 (Dec. 1994), 334-356.

**25**

ZHANG, L., DEERING, S., ESTRIN, D., SHENKER, S., AND ZAPPALA, D. RSVP: A new resource ReSerVation Protocol. *IEEE Network* (Sept. 1993).

**Author Biography**

Jason Leigh is a PhD candidate at the Electronic Visualization Laboratory at the University of Illinois at Chicago. CAVERNsoft is his PhD dissertation.

Andrew E. Johnson is an assistant professor at the University of Illinois at Chicago. His research interests include collaborative virtual environments and using virtual reality as a high level interface for visualizing large scientific datastores.