Proc. of the Symposium on Virtual Reality Software and Technology 2000, Oct 22-25, 2000, Seoul, Korea

CAVERNsoft G2: A Toolkit for High Performance Tele-Immersive Collaboration

Kyoung S. Park, Yong J. Cho, Naveen K. Krishnaprasad, Chris Scharver, Michael J. Lewis,
Jason Leigh, Andrew E. Johnson
Electronic Visualization Laboratory
851 S. Morgan St., Room 1120 (M/C 154)
University of Illinois at Chicago
Chicago, IL 60607, USA
1 312 996 3002
cavern@evl.uic.edu

ABSTRACT

This paper describes the design and implementation of CAVERNsoft G2, a toolkit for building collaborative virtual reality applications. G2's special emphasis is on providing the tools to support high-performance computing and data intensive systems that are coupled to collaborative, immersive environments.

This paper describes G2's broad range of services, and demonstrates how they are currently being used in a collaborative volume visualization application.

Keywords

Tele-immersion, high-performance computing, data-mining, networking library, VR, CVE.

1. INTRODUCTION

Tele-immersion is defined as the synthesis of collaborative virtual environments (CVEs), audio and video conferencing, and supercomputing resources and massive data stores, all interconnected and running over high-speed national or worldwide networks [10]. Tele-immersion enables participants in distant locations to collaborate in a shared environment as if they are in the same room. The development of tele-immersive applications is considered one of the most challenging areas of research in VR as it requires expertise in VR, networking, database management, collaboration and high performance computing.

CAVERN, the CAVE Research Network, is an alliance of industrial and research institutions equipped with CAVE® [4] and ImmersaDesk® VR systems, and high-performance computing resources. High-speed national and international networks peering at STAR TAP (the Science, Technology and Research Transit Access Point [5]) are able to support tele-immersive engineering, design, education and training, scientific visualization, and computational steering.

Our goal is to make these systems convenient, so that scientists and designers can do real work within these shared environments without having to worry about how the collaboration is sustained.

CAVERNsoft G1 [13] was our previous attempt to provide a system to rapidly generate tele-immersive applications, and retrofit a legacy of non-collaborative VR applications, with tele-immersive capabilities. Other similar toolkits available, academically or commercially, including DIVE [2], NPSNET[16], WTK/WorldUp/World2World [18][19][20], BrickNet [21], Avango [24], and Bamboo [26]. The primary difference between CAVERNsoft and the others is its focus on integrating collaborative VR with high-performance and data-intensive computing over high-speed networks.

These applications typically have the following additional requirements beyond those expected of CVEs:

- Support for 64-bit precision
- Support for 64-bit memory addressing
- Support for 64-bit file access
- Support for rapid transmission of massive data files
- Support for performance monitoring

This paper will begin with an historical overview, and what motivated this second generation design of CAVERNsoft. Following this will be a description of G2's capabilities, with particular attention given to G2's data distribution and network performance monitoring capabilities. Finally this paper will describe an application built using G2 as an example of how application programmers might make use of G2's capabilities.

LEAVE THIS TEXT BOX IN PLACE
AND BLANK

2. THE HISTORY OF CAVERNsoft

The concept behind CAVERNsoft began with the development of an application called CASA for an electronic visualization event held at EVL in May 1995 [3]. CASA (Computer Augmentation for Smart Architectonics) was a proof of concept to illustrate how virtual environments could be used to prototype smart environments. In 1995, CASA demonstrated the earliest application of collaborative VR using the CAVE. To facilitate this, a shared variable model of a distributed shared memory system (DSM), which uses a reliable protocol, was developed to eliminate the need of programmers to develop specific protocols for network communication. This was also the first instance of a DSM being used to build collaborative CAVE applications.

Expanding on the CASA work, CALVIN (Collaborative Architectural Layout Via Immersive Navigation) was developed in October 1995 [12]. CALVIN allowed collaborating users to edit an architectural design by picking up objects, moving, rotating and scaling them. CALVIN extended CASA's DSM system by also providing persistence. Using CALVIN, collaborators could save versions of their design and resume their work at a later time.

Through the development of CALVIN and its subsequent demonstrations at conference events, two issues became apparent: 1. The use of reliable TCP to deliver all of the DSM data introduced too great a network lag in the application; 2. The lack of an event-driven system to notify a client when a shared variable had been updated meant that each variable had to be polled to determine if a change had occurred. Hence, it appeared that a message passing system might have been preferable. Also, we were beginning to build a new application called NICE (Narrative Immersive Collaborative/Constructionist Environment) [11].

NICE was an educational environment that allowed children to collaboratively plant virtual vegetables in a virtual garden. A central simulation server was used to simulate the growth of the vegetables based on the conditions in the garden. In NICE we attempted to use a purely message passing system to share all networked data. A UDP connection was used to share avatar state information and TCP connections were used to share the state of the garden as well as to allow the download of avatar 3D models from an HTTP server. While the message passing system used in NICE allowed the application developer to control networking more precisely, data distribution was no longer transparent. Furthermore, whereas in a DSM model, late joiners would find out about the state of the world by contacting a central server, in a message-passing model, late joiners had to listen for incoming messages to gradually construct the state of the world.

To address these issues, and build a more robust and reusable architecture for data sharing in CVEs, we began designing what is now known as CAVERNsoft.

CAVERNsoft's initial design criteria included:

- Allowing the use of multiple protocols such as TCP, UDP, multicast and HTTP
- Supporting both a distributed shared memory and a message passing model
- Providing persistence
- Providing an event generation and handling mechanism
- Providing low-level access to networking calls as well as high-level abstractions
- Facilitate the construction of new applications and the retrofitting of legacy applications

To test the viability of these ideas, we developed a proof of concept (CAVERNsoft G0) in September 1996. Additional capabilities such as greater reliability and the ability to share data over a variety of network connections simultaneously, culminated in the first release of CAVERNsoft (G1) in September 1997 [13]. G1 was unique in a number of ways. Firstly, it broke the traditional model of DSMs by allowing clients to share information transparently over both reliable and unreliable links. Secondly, G1 was client and server symmetric. That is, a client program was indistinguishable from a server program. Clients could transparently connect to other clients to access their DSM over a customizable network connection. This flexibility allowed users to build any topology of collaborative applications, whether it was a star with a centralized server; or a chain with each client linking to one neighbor; or completely decentralized over multicast.

G1 also went further to include higher-level abstractions that would provide C++ classes for sharing 3D avatars and audio streaming. These abstractions were structured in an application shell called LIMBO [14] to facilitate the development of new collaborative VR applications. LIMBO not only provided shared avatars, but also a means to persistently share 3D models and their state information.

Approximately a dozen collaborative applications were built using G1 before it was finally replaced with G2 in September 1999 [15]. The remainder of this paper will discuss G2's components and an example of how they are used to build a tele-immersive volume visualization application.

3. DESIGN AND IMPLEMENTATION

G1 and G2 are both C++ libraries. The major drawback of G1 was that it was a large monolithic system, which made the inclusion of new capabilities difficult. It was also heavily threaded using Nexus and Globus [7], thus causing

incompatibility with other graphical or VR libraries. Globus is a C toolkit developed by Argonne National Laboratory for building high-performance computing applications and coordinating their resources.

CAVERNsoft G2 took the approach of separating many of the data delivery mechanisms in G1, into smaller independent classes or modules (Figure 1.) Hence G2 is a lightweight toolkit rather than a monolithic system. As a toolkit, G2 comprises low-level modules to provide full control of networking at the socket level; middle-level data distribution modules such as remote procedure calls; and high-level modules such as application shells and classes for rendering avatars using the CAVE library and IRIS Performer. Note that the low- and mid-level modules can be used independently of the CAVE library or IRIS Performer, and hence can be used to build cross-platform client/server applications.

3.1 Low-level Modules

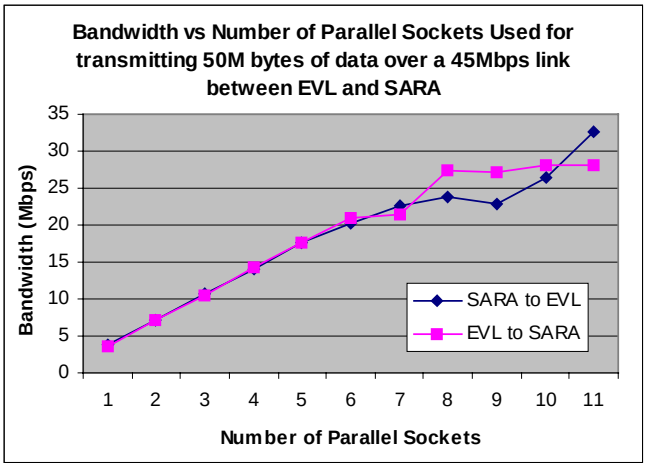
The low-level network modules provide socket-level networking classes for TCP, UDP and multicast communications, and cross-platform numerical data conversion. Building modules, or applications on top of these modules, guarantees their portability across platforms.

3.1.1 CAVERNnet_tcp_c, CAVERNnet_udp_c, CAVERNnet_mcast_c classes

CAVERNsoft provides simple classes to support TCP, UDP and multicast protocols. UDP is frequently used to deliver avatar tracker data as the loss of a packet is soon followed by another [16].

3.1.2 CAVERNts_condition_c, CAVERNts_mutex_c, CAVERNts_thread_c classes

G2 also provides low-level modules that encapsulate thread management and mutual exclusion capabilities of different operating systems. Support is provided for POSIX th2.1(t)C(3.1.14.1( )-o)-6.9d.5(s)8.6.3(V) .9(usd)-5.9(ct)6.1(s4.1( )ene)7tW(m)-19.c e



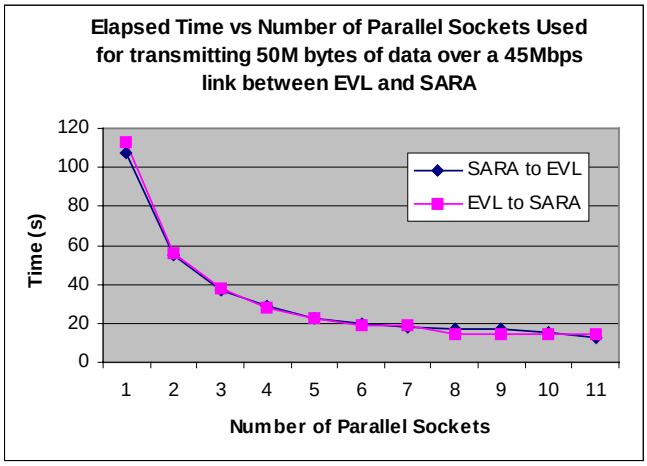


Figure 2 Results of a parallel socket test between EVL (Chicago, USA) and SARA (Amsterdam, Netherlands) over STAR TAP.

3.2.3 CAVERNnet_parallelTcp_c class

The main networking requirement in high-performance distributed computing applications is the need to rapidly move massive amounts of data from one site to another. While high bandwidth networks already connect many members of CAVERN, these networks are still, for the most part, incapable of providing Quality of Service (the ability to establish desired application bandwidth, latency and jitter.) Furthermore, as these CAVERN users are typically distributed around the world (for example between Chicago and Amsterdam) the network latency (approximately 95ms) between sites imposes what is called the "long fat network" (LFN) problem [22]. The LFN problem occurs because in most cases the kernel's TCP window size is set too low to take advantage of the available network capacity. TCP windows specify how much information a TCP socket may send before waiting for an acknowledgement. For long distance networks this often results in severe bandwidth under-utilization. Hence while a client may have a 45Mbps connection to a remote site (as is the case between Chicago and Amsterdam) it may not be able to access more than 5Mbps at a time. Increasing the TCP window size often solves this. However there remain legacy versions of TCP that use a 16-bit window size specification and cannot increase their window size beyond 64Kbytes.

G2 provides a temporary solution by providing a parallel socket class. This class works like G2's regular TCP socket class except a data buffer is partitioned and transmitted over several sockets rather than just one. This technique is often known as socket striping. Again with our example from Chicago to Amsterdam, Figure 2 shows the performance of parallel sockets when transmitting a 50Mbyte file using between 1 and 11 sockets. Notice bandwidth utilization improves dramatically from 4Mbps to 32Mbps.

3.2.4 CAVERNnet_extendedTcp_c,CAVERNnet_extendedParallelTcp_c classes

Another useful middle level module G2 provides is the Extended TCP module, which combines G2's data packing and unpacking routines with its TCP class. The Extended *Parallel* TCP module provides the same data conversion enhancements for parallel sockets.

3.2.5 CAVERNnet_remoteFileIO32_c, CAVERNnet_remoteFileIO64_c, CAVERNnet_remoteParallelFileIO32_c, CAVERNnet_remoteParallelFileIO64_c classes

The Remote File I/O modules provide the capability for uploading and downloading files from a remote server. The provision of both 32-bit and 64-bit versions, as well as parallel socket versions of the class allows for the efficient delivery of all file sizes, including those larger than 2 Gigabytes. The 64-bit version effectively allows one to deliver Terabyte files.

A Terabyte file will take approximately a day to deliver on a 100Mbps network link, assuming full utilization of the capacity. In these cases the need for parallel TCP becomes even more prominent.

3.2.6 CAVERNdb_c, CAVERNmisc_observer_c and subject_c classes

G2 provides distributed shared memory emulation via the CAVERNdb (or database) class. This is essentially a client/server database with default data reflection. Hence any updates to the database are propagated to all subscribers of the database. Clients are notified by either a traditional callback function or a subject/observer mechanism [8]. The subject/observer mechanism is essentially an object-oriented replacement for callbacks. The subject maintains a list of its observers for specific events and each observer will be triggered whenever the specific event occurs.

The database assumes a Unix-like directory hierarchy with the leaf nodes containing the individual data values. These data values are intended to be small to expedite state information sharing rather than bulk data sharing.

3.2.7 CAVERNnet_rpc_c class

To compliment G2's DSM and message passing capabilities, remote procedure calling is also provided. This allows clients and servers to invoke each other's functions and procedures. This is a widely used technique for distributed computing, however, it has been found to be less applicable in real-time CVE applications.

3.3 High-level Developer Modules

The high-level modules in G2 provide functionality for audio streaming, avatars, speech recognition, 3D menus and other graphical modules utilizing the CAVELib™ and IRIS Performer libraries.

3.3.1 CAVERN_audioStream_c class

The audio streaming module relies on a UDP reflector to share voice or audio between clients. Audio can be shared from 8kHz to 44kHz, bandwidth permitting. Threshold is also provided as a means to reduce bandwidth utilization when audio levels fall below user-definable amplitude.

3.3.2 CAVERN_baseAvatar_c, CAVERN_perfArticulatedAvatar_c classes

The base avatar module provides a non-graphical implementation of avatars. This is useful as a starting point for developing graphical avatars as it encapsulates the basic information sharing requirements. For example, G2 uses this base module to provide a CAVE/Performer avatar class that supports a single head tracker and two wand trackers. Inverse kinematic calculations are performed on the wand trackers to provide articulated arms for the avatars.

3.3.3 CAVERN_perfCAVENavCollision_c class

As in many other VR libraries, G2 provides facilities for navigation and collision detection within a CVE. This module employs the CAVE library and Performer.

3.3.4 pfNetDCS_c class

The networked DCS (dynamic coordinate system) module is an extension of Performer's DCS node such that transformations within the DCS are automatically shared with other collaborators over the network. The advantage is that programmers may manipulate networked objects in the same way they manipulate standard Performer objects. A similar approach has been taken in Avango [24].

3.3.5 CAVERN_perfCAVEPickNMove_c class

Combining the pfNetDCS_c module with the CAVE library the pick and move module allows programmers to easily write code to pick, move and orient objects in a shared manner.

3.3.6 CAVERN_perfGui_c

The Performer-based menu module creates push-to-select menus within the virtual environment. This module loads model files to represent its "buttons" and hence its appearance is customizable. Since the module uses Performer for model loading, any models supported by Performer can be used, such as pfb, flt, iv, dxf, obj, 3ds.

3.3.7 The Collaborative Animator

The collaborative animation module is actually a complete running program. It allows collaborators to load a series of models into the environment and cycle through them rapidly to form a flipbook animation. The application provides avatar support; the ability to turn wire frame meshing on; and to push a cutting plane through the scene. All capabilities are collaboratively shared hence one participant may advance the flipbook while another is manipulating the cutting plane.

3.3.8 LIMBO

Similar to the Collaborative Animator, LIMBO provides a more generic program shell for jumpstarting application development. LIMBO takes the aforementioned modules for avatar presentation, shared object manipulation, and collision detection, and organizes them into a simple extensible framework. This is the first imposition of a framework in G2, and is primarily intended for the development of new applications. One of G2's chief design goals has been to develop a system that allows both the construction of new collaborative applications as well as the retrofitting of legacy applications. Legacy applications often have an existing program structure in place that is difficult to reorganize into a new framework. G2's toolkit approach allows one to insert collaborative capabilities into the program as needed, without requiring the program to adopt a new framework. Using this approach, it has been quite effortless to insert collaborative capabilities into General Motors' VisualEyes automobile styling system [15].

4. PERFORMANCE MONITORING AND ANALYSIS

High performance computing applications often consume enormous amounts of computational, network and storage resources. These resources are often limited, hence tuning of these applications, as well as their visualization clients, is important. G2 supports this by imbedding performance monitoring routines in all the network modules. The performance calculations are done automatically whenever the network is used and the statistics are accessible via member functions in each of the networking classes. These statistics conform to the Netlogger [17] format, which is simply a list of <label, value> pairs. The following is an example:

TIME=955664372.441101 SELF_IP=131.193.48.163
REMOTE_IP=131.193.48.164 SELF_PORT=9977
REMOTE_PORT=1811
STREAM_INFO=131.193.48.164_AVATAR_SERVER
COMMENT=TRACKER_UDP MIN_LAT=0.000318
AVG_LAT=0.002742 MAX_LAT=0.069904
INST_LAT=0.000734 JITTER=0.001046 MIN_IMD=0.000093
AVG_IMD=1.902742 MAX_IMD=480.669540

INST_IMD=0.000715 AVG_RBW=11.573473
INST_RBW=44754.160720 AVG_SBW=401.712774
INST_SBW=733308.778808 BURSTINESS=30118.629172
TOTAL_READ=8258 TOTAL_SENT=306604
PACKETS_READ=376 PACKETS_SENT=7860

The parameters measured are Minimum Latency (MIN_LAT), Average Latency (AVG_LAT), Maximum Latency (MAX_LAT), Instantaneous Latency (INST_LAT), Jitter (JITTER), Minimum Inter-Message Delay (MIN_IMD), Average Inter-Message Delay (AVG_IMD), Maximum Inter-Message Delay (MAX_IMD), Instantaneous Inter-Message Delay (INST_IMD), Average Read Bandwidth (AVG_RBW), Average Send Bandwidth (AVG_SBW), Burstiness (BURSTINESS), Throughput at the receiving end (TOTAL_READ), Throughput at the transmitting end (TOTAL_SEND), Number of Packets read (PACKETS_READ) and Number of Packets sent (PACKETS_SENT).

A brief account of the parameters measured, along with their units of measurement and the formulae used, is given as follows:

- **Latency (one-way) = T_s - T_r**

  Where T_s is the Time recorded at the sender's end and T_r is the time recorded at the receiving end (both ends should be time synchronized). The unit of measurement is seconds. The minimum (MIN_LAT), average (AVG_LAT), maximum (MAX_LAT) and instantaneous (INST_LAT) latencies are calculated and provided for further analysis.

- **Jitter** = **E [{ L_i – E[L] }]**

  Where E is the Expectation of a data set, L is the set of 100 most recent instantaneous latency samples and L_i is the instantaneous latency. The unit of measurement is seconds.

- **Inter Message Delay** = **T_{i+1} - T_i**

  Where T_i and T_{i+1} are instances of two consecutive messages received. The unit of measurement is seconds. Like latency, the minimum (MIN_IMD), average (AVG_IMD), maximum (MAX_IMD) and instantaneous (INST_IMD) inter message delays are calculated and recorded.

- **Bandwidth** = [$\delta d / \delta t$]

  Where δd is the data in bytes, received/sent over a time δt. The unit of bandwidth is bytes/sec. AVG_RBW, INST_RBW, AVG_SBW and INST_SBW represents the average and instantaneous values of read and send bandwidth, respectively.

- **Burstiness** = **E [{ B_i – E[B] }]**

  Where E is the Expectation of a data set, B is the set of 100 most recent instantaneous bandwidth samples and B_i is the instantaneous read bandwidth. The unit of measurement as in bandwidth is bytes/sec.

In addition to being able to retrieve these statistics via member functions in each of the network classes, these statistics can be streamed in real-time to a network performance daemon. A CAVE-based network visualization client called QoSIMoto (QoS Internet Monitoring Tool) [25] can then be connected to the daemon to watch, in real-time, the bandwidth, latency and jitter of a collection of simultaneous network flows. This allows us to find correlations between application events and network utilization.

5. A TELE-IMMERSIVE APPLICATION

In this section we will describe one application of G2 - others exist, however within the limits of this paper we will describe one in some detail.

The Collaborative Image Based Rendering Viewer (or CIBRView) is a tool designed to address two problems related to volume visualization. Firstly, it attempts to make volume visualization of extremely large animated data volumes possible on both low-end desktop workstations and high-performance graphics systems, such as the ImmersaDesk and CAVE. Secondly, CIBRView attempts to allow free-form collaborative viewing of these images, so that multiple participants at remote sites and on heterogeneous visualization platforms may work together to interpret massive data sets.

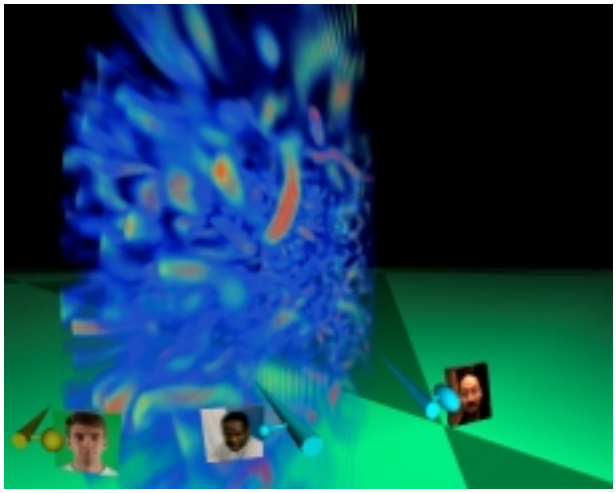


Figure 3 Collaborators in a CIBR View session.

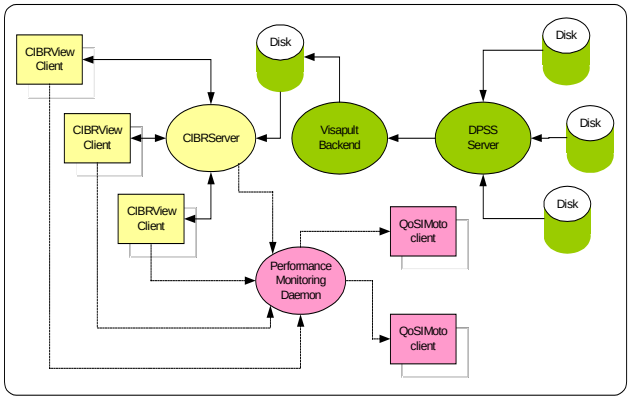


Figure 4 Connectivity of a typical performance monitored CIBRView collaborative session.

This is achieved by connecting collaborating visualization clients to a supercomputing backend called Visapult [1]. Visapult was developed by Wes Bethel at Lawrence Berkeley Laboratory, to generate semi-translucent composite image slices of a data volume using multiple processors of an SGI Onyx or Origin class supercomputer. Data volumes tend to be large. The volumes that we have been working with thus far are on the order of approximately 40Gigabytes. These files are typically housed on large disks at Lawrence Berkeley Lab and made accessible by a Distributed Parallel Storage System (DPSS) server [23].

Within the collaborative environment, participants are represented virtually by their name badges or photos attached to a 3D pointer they could use to point to items of interest. While a desktop user would use a mouse to operate in the environment, a CAVE user would use a 3D wand. Participants are able speak to each other via a shared audio stream, while they collaboratively cycle through sequences of animation frames, or slice through the data with a cutting plane (Figure 3).

Figure 4 shows the network connectivity between CIBRView, Visapult, the DPSS server, the performance monitoring daemon and two QoSIMoto clients. The DPSS server reads volume data rapidly from parallel disks and streams them to the Visapult backend for rendering. The rendered slices are stored to a local disk accessible to a CIBRView server. This CIBRView server maintains the state information for the CIBRView virtual environment. Upon generation of each image slice, the remote CIBRView clients request copies of the slice via the CIBRView server. The CIBRView server consists of a G2 remote file I/O server; a TCP reflector to reflect information from newly joining avatars; a UDP reflector to reflect avatar tracking data; and a CAVERNdb server to share the state of the cutting plane, the next available image slice and the current animation frame being viewed by all the clients. The CIBRView client possesses corresponding client interfaces to the CIBRView server's modules. Performance data generated by the CIBRServer and the CIBRView clients are streamed to a performance-monitoring daemon, which is then distributed to all listening QoSIMoto clients. We have found the ability to monitor network utilization in real-time to be useful, as it allows us to correlate events in the CVE with its impact on the networks, and vice versa.

6. CONCLUSION AND FUTURE WORK

We have described the motivation and design of CAVERNsoft G2, a cross-platform C++ toolkit for building collaborative tele-immersive environments.

In our continuing work to optimize throughput between high performance computing systems and CVEs, we are experimenting with two network Quality of Service testbeds: GEMnet, a Chicago-Tokyo network that employs a proprietary Integrated Servicecs QoS implementation called Media Cruising Signaling Protocol (developed by Sony and Nippon Telephone and Telegraph); and EMERGE, a national Differentiated Services test bed supported by the U.S. Department of Energy. We are studying these testbeds to determine how well they are able to make bandwidth and jitter guarantees.

Furthermore we are exploring techniques for low-latency reliable transmission of state or streaming information via error correcting schemes [6]. The goal is to provide a protocol with the low latency of UDP, but with the reliability of TCP.

Additional details about CAVERNsoft G2, its capabilities, the implementation as well as the distribution are available on the web at http://www.evl.uic.edu/cavern/cavernG2/.

7. ACKNOWLEDGMENTS

We would like to thank Stichting Academisch Rekencentrum Amsterdam (SARA) in the Netherlands for assisting us in performing our parallel socket experiments. We would also like to thank Stuart Bailey and Robert Grossman of the National Center for Data Mining for sharing their parallel socket algorithm.

The virtual reality research, collaborations, and outreach programs at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago are made possible by major funding from the National Science Foundation (NSF), awards EIA-9802090, EIA-9871058, ANI-9980480, ANI-9730202, and ACI-9418068, as well as NSF Partnerships for Advanced Computational Infrastructure (PACI) cooperative agreement ACI-9619019 to the National Computational Science Alliance. EVL also receives major funding from the US Department of Energy (DOE), awards 99ER25388 and 99ER25405, as well as support from the DOE's Accelerated Strategic Computing Initiative (ASCI) Data and Visualization Corridor program. In addition, EVL receives funding from Pacific Interface on behalf of NTT Optical Network Systems Laboratory in Japan and Microsoft Corporation.

University of Illinois.

8. REFERENCES

[1] Bethel, W. Visapult web site, LBNL, http://vis.lbl.gov/projects/visapult/index.html

[2] Carlsson, C. and Hagsand, O., "DIVE – A Multi-User Virtual Reality System," *Proc. of IEEE VRAIS '93*, Seattle, WA, September 18-22, 1993, pp. 394-400.

[3] CASA (Computer Augmentation for Smart Architectonics) web site, Electronic Visualization Event 4 (EVE4), EVL, University of Illinois at Chicago, May 9-19, 1995, http://www.evl.uic.edu/spiff/casa/

[4] Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V. and Hart, J. C., "The Cave Automatic Virtual Environment," *Communications of the ACM*, 35(2):64-72, June 1992.

[5] DeFanti, T. A. and Goldstein, S. STAR TAP web site, http://www.startap.net/

[6] Fang, R., Schonfeld, D., Rashid, A., Leigh, J., "Forward Error Correction for Multimedia and Tele-immersion Streams," EVL technical report, February 2000. http://www.startap.net/images/RayFangFEC1999.pdf

[7] Foster, I. and Kesselman, C., "Globus. A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputing Application*, 11(2):115-128, 1997.

[8] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns – Elements of Resuable Object-Oriented Software*, pages 293-303, Reading, MA: Addison-Wesley, 1995.

[9] Hagsand, O., Lea, R., Stenius, M., "Using Spatial Techniques to Decrease Message Passing in a Distributed VE System," *Proc. of VRML '97 – Second Symposium on the Virtual Reality Modeling Language*, Monterey, California, February 24-26, 1997, pp. 7-16.

[10] Johnson, A. E., Leigh, J., and DeFanti T., "Multi-Disciplinary Experiences with CAVERNsoft Tele-Immersive Applications," *Proc. of Fourth International Conference on Virtual System and Multimedia*, November 1998, pp. 498-503.

[11] Johnson, A., Roussos, M., Leigh, J., Barnes, C., Vasilakis, C., Moher, T., "The NICE Project: Learning Together in a Virtual World," *Proc. of IEEE VRAIS '98*, Atlanta, Georgia, March 14-18, 1998, pp.176-183.

[12] Leigh, J., Johnson, A., Vasilakis, C., DeFanti, T., "Multi-perspective Collaborative Design in Persistent Networked Virtual Environments," *Proc. of IEEE VRAIS '96*, Santa Clara, CA, March 20 - April 3, 1996, pp. 253-260, 271-272.

[13] Leigh, J., Johnson, A. E. and DeFanti, T. A., "CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments," *Journal of Virtual Reality Research, Development and Applications*, 2(2):217-237, 1997.

[14] Leigh, J., Rajlich, P., Stein, R., Johnson, A. E., DeFanti T. A., "LIMBO/VTK: A Tool for Rapid Tele-Immersive Visualization," *CDROM proc. of IEEE Visualizaton '98*, Research Triangle Park, NC, October 18-23, 1998.

[15] Leigh, J., Johnson, A., DeFanti, T., et al., "A Review of Tele-Immersive Applications in the CAVE Research Network," *Proc. of IEEE VR '99*, Houston TX, March 13-17, 1999.

[16] Macedonia, M. R., Zyda, M. J., Pratt, D.R., Barham, P.T., and Zeswitz, S., "NPSNET: A Network Software Architecture for Large-Scale Virtual Environments," *Presence*, 3 (4): 265-287, 1994.

[17] Netlogger web site, http://www.didc.lbl.gov/NetLogger/homepage.html

[18] Sense8 Corp. WorldToolKit Reference Manual, Release 8.

[19] Sense8 Corp.WorldUp Users Guide, Release 4.

[20] Sense8 Corpo. World2World Release 1. Tech. Overview.

[21] Singh, G., Serra, L., Png, W., Wong, A., and Ng, H., "BrickNet: Sharing Object Behaviors on the Net," *Proc. IEEE VRAIS '95*, Research Triangle Park, NC, March 11-15, 1995, pp. 19-25.

[22] Stevens, W. R., *TCP/IP Illustrated Vol.1*, Chapter24, pp. 344–350.Addison Wesley, 2nd Edition, 1994.

[23] Tierney, B.L., Distributed Parallel Storage System, http://www-didc.lbl.gov/DPSS/

[24] Tramberend, H., "Avocado: A Distributed Virtual Reality Framework," *Proc. of IEEE VR '99*, Houston, TX, March 13-17 1999, pp. 14-21.

[25] QoS Internet Monitoring Tool (QoSIMoTo) web site, http://www.evl.uic.edu/cavern/qosimoto/

[26] Watsen, K. and Zyda, M., "Bamboo – A Portable System for Dynamically Extensible, Real-Time, Networked Virtual Environments," *Proc. of IEEE VRAIS '98*, Atlanta, GA, March 14-18, 1998, pp. 252-259.