

# Collaborative Visualization and Real-time Monitoring of Scientific Data (Submission ID: sketches\_0133)

Amin Ghadersohi<sup>\*†</sup>      Dave Pape<sup>‡</sup>      Charles M. Weeks<sup>§</sup>      Mark L. Green<sup>\*</sup>      Russ Miller<sup>\*\*§</sup>  
ag28@buffalo.edu      depape@buffalo.edu      weeks@hwi.buffalo.edu      mlgreen@ccr.buffalo.edu      miller@buffalo.edu

<sup>\*</sup>Center for Computational Research, <sup>†</sup>Dept. of Comp. Sci. & Eng., <sup>‡</sup>Dept. of Media Study, SUNY-Buffalo

<sup>§</sup>Hauptman-Woodward Medical Research Institute, Buffalo NY

Networked visualization frameworks that support collaborative work have become increasingly important in many application areas in science and engineering. Especially support for heterogeneous display environments, ranging from desktop systems or even PDAs to fully-immersive environments such as the CAVE. In a world where the term collaborative visualization is usually associated with Java™-based applet architectures or multi-user VNC sessions, we have designed an object-oriented architecture for delivering consistent, low-latency, real-time data to geographically disparate visualization clients. Applications for such a system extend to bioinformatics, military simulation and training, collaborative design, persistent multi-user worlds, online communities and education.

Challenges with the networking and distribution of the data include network bandwidth, heterogeneous distributed interaction, failure management, and scalability. We demonstrate proof-of-concept by implementing the architecture to visualize protein structure data from a critical, complex, and computationally intensive application from structural biology (*Shake-and-Bake*) that was recently listed on the IEEE poster of “Top Algorithms of the 20<sup>th</sup> Century”. The system enables distributed visualization of large 3D protein structures across multiple platforms and graphical environments for purposes of interactive monitoring and modification. It addresses collaboration in the context of distributed visualization for large protein data.

To achieve the brute high performance that is demanded of any Collaborative Virtual Environment (CVE), our system is implemented using the C++ language and OpenGL API. However, there are many design issues, more important than the implementing language, which we have to consider. We address network topologies, scalability, space structuring, data consistency, user interaction and representation.

One of the two network topologies that we consider is the centralized model where one computer (database) collects all data and sends updates to the users. Although this design has a simple structure, it is by no means scalable, since the database becomes a bottleneck when it has to send updates to many clients. The second network model is a distributed model where each client maintains its own copy of the database. Each client takes care of sending updates to other clients. Without any central control, this design makes the task of data synchronization and integrity very challenging, and by placing the burden on the network we still have to deal with bottleneck issues. Therefore, we will adopt a topology that has both the advantages of the centralized and distributed models. For instance, clients that get disconnected due to network problems may re-connect and re-sync their data, similar to ad hoc database systems.

The concept of scalability directly leads to space structuring, which is addressed by a multi-server design similar to how cellular networks are divided into hexagonal cells. It is also

possible to have different sized cells and user-centered dynamic structures. This scheme is more scalable than the two network topologies discussed earlier, leaving the issue of assigning a server to a client. However, the main concern of the network still is to avoid performance degradations, due to bottlenecks, when dealing with large datasets such as protein molecules with hundreds of thousands of atoms. Moreover, with the possibility of network lags, we need to address the issue of data consistency and integrity when two or more users simultaneously modify the same data.

Clearly, there is a tradeoff that needs to be made between consistency and throughput. It is impossible to allow dynamic shared state to change frequently and guarantee that all host simultaneously access identical versions of that state. However, we can guarantee absolute consistency by forcing clients to wait for others to acknowledge changes. The issue here is considering latency and available bandwidth; this may imply slowing down too much. In another scheme the clients send out updates as soon as changes are made, but some users will receive the updates quickly and others will not, resulting in inconsistent views.

Since we are dealing with scientific data, we need to provide capabilities to guarantee absolute consistency when it is needed. This is achieved through locking and access management techniques. However, for much of the information that is exchanged by the network, such as relative positions of users in the virtual environment, we can use techniques such as dead-reckoning to derive estimated values. Instead of sending updates at frame rate, send parameters that describe the trajectory of the object (example: initial position and velocity); each participant displays the trajectory at its own rate. Although this has the advantage of reduced bandwidth usage, more cycles are needed to compute trajectory position at each client. Another problem is how to converge the predicted position with a new position update that may arrive from the network. A solution is to send aggregate updates by only sending when positions change more than a certain threshold.

Other issues include user interaction, representation and communication. The heterogeneous nature of the clients, led us to design an API for extending the architecture to many graphics environments and input devices - currently GLUT and CAVE. The problem of user representation is how users perceive each other in the virtual environment. Avatars - 3D models - can be used to signify the users in the environment. Finally, users may want to communicate with each other vocally, or textually.

Our system uses advanced communication models, such as multiple-servers, message broadcasting and IP Multicast, and better database models that take advantage of algorithms and properties of hierarchal data to reduce communications. It addresses the overall issue of keeping the balance between the high performance and reliability that is expected from a state-of-the-art collaborative virtual environment.