Transport Layer for Remote Client-Server Rendering Systems

Arun Kumar Paruchuri, Joerg Meyer
Mississippi State University, Department of Computer Science

Abstract

Generally, medical visualization systems are not used by only one person. Therefore, it is necessary that a group of people works together on a diagnosis for a certain medical condition, or to achieve a common research objective. The World Wide Web (WWW) can be used to allow the simultaneous analysis of data by collaborators at remote sites. Moving large data sets from data repositories, such as the HPSS (High-performance Storage System) at San Diego Supercomputer Center (SDSC), is extremely time-consuming, and the remote user usually has limited rendering and processing resources. Therefore, it is best to leave the data on the server, and to use visualization resources at the local site only for reassembling of data blocks and post-processing. The stratgies used to break data into smaller bricks and compress those bricks for faster transmission have been published earlier [4]. This framework describes the transport layer, which allows us to implement a client/server model for communicating with each other using standardized Internet protocols (Figure 1).

Fig. 1: Texture-based Java3D rendering client and server

Fig. 2: Data brick extraction on the server (cross-section)

The user selects a 3-D region of interest in a 2-D projection of a 3-D object using the mouse and initiates a request. The request is handled through a Java-based web interface. The request travels over the network to the server, which interprets the request and accesses one or several data bricks (sub-volumes) from a large data set. The data set can be several gigabytes, which is more than the amount of data that can be transfered over the network in interactive time frames (Figure 2). The client reassembles the data bricks, puts them in the correct order for rendering (back-to-front projection, texture-based Java3D rendering client). Finally, the rendering is done on the client side. A low-resolution representation of the entire data set (cryo-section of a human brain, 3.57 GB, Figure 2) is used as context information, while the region of interest is rendered at a higher resolution (3-D images [3]). Image refinement can be done easily by swapping the textures on a set of perpendicular planes using the data sent by the server.

The actual multi-threaded communication uses TCP as the underlying connection protocol. Once a socket connection has been established between the client and the server, the data flows in both directions (client sends request, server responds by sending data bricks). Our implementation extends the functionality of the standard byte transfer protocol to a simple interface for transmitting compressed sub-volumes and corresponding location IDs.

The advantage of this approach is low requirements for the hardware and software on the client's machine. In this model, the server does almost all the work, and just the sub-volumes are sent to the client for final rendering.

(Data set courtesy of Arthur W. Toga's group, Department of Neurology, UCLA School of Medicine, Los Angeles.)

References

[1] Stevens, W. Richard. Unix network programming. Addison Wesley Longman Singapore, 2000.

[2] Tanenbaum, Andrew S. Computer networks. New Delhi: Prentice Hall of India, 1997.

[3] Takanashi, Ikuko, Eric B. Lum, Meyer, Joerg, Kwan-Liu Ma, Bernd Hamann, and Arthur J. Olson. Segmentation and Volume Rendering of Human Brain Cryosections (Abstract). IEEE Visualization 2000, Work in Progress (CD-ROM). Salt Lake City, Utah, October 8 - 13, 2000.

[4] Meyer, Joerg, Ragnar Borg, Bernd Hamann, Kenneth I. Joy, and Arthur J. Olson. VR-based Rendering Techniques for Large-scale Biomedical Data Sets. Online Proceedings of NSF/DoE Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization. Granlibakken Conference Center, Tahoe City, CA, pp. 73 - 76, October 15 - 17, 2000.

[5] Networking and multi-threading functions.
http://qdn.qnx.com/support/docs/neutrino_qrp/lib_ref/summary.html#CLASSIFICATION