TeraVision: a Distributed, Scalable, High Resolution Graphics Streaming System

Rajvikram Singh+, Byungil Jeong, Luc Renambot, Andrew Johnson, Jason Leigh
Electronic Visualization Laboratory
University of Illinois at Chicago
http://www.evl.uic.edu/cavern
+rsingh@evl.uic.edu

Abstract

In electronically mediated distance collaborations involving scientific data, there is often the need to stream the graphical output of individual computers or entire visualization clusters to remote displays. This paper presents TeraVision as a scalable platform-independent solution which is capable of transmitting multiple synchronized high-resolution video streams between single workstations and/or clusters without requiring any modifications to be made to the source or destination machines. Issues addressed include: how to synchronize individual video streams to form a single larger stream; how to scale and route streams generated by an array of MxN nodes to fit a XxY display; and how TeraVision exploits a variety of transport protocols. Results from experiments conducted over gigabit local-area networks and wide-area networks (between Chicago and Amsterdam), are presented. Finally, we propose the Scalable Adaptive Graphics Environment (SAGE) - an architecture to support future collaborative visualization environments with potentially billions of pixels.

1. Introduction

One of the common problems faced in Amplified Collaboration Environments [5] and scientific visualization applications is termed as the 'Display Docking' or the 'Display Pushing' problem. It expresses the need for distributing visualizations or presentations generated on one or more computers, to remote sites for viewing and/or post-processing. A typical source of graphics or video in such a case could be computers ranging from laptops showing presentations, to compute clusters number crunching terabytes of data and rendering high-resolution visualizations on tiled displays.

A typical scientific visualization pipeline consists of four stages: storage, computation, rendering and display. The storage stage would typically store data in the order of several gigabytes, even terabytes. The computation (or processing) phase reduces the data to its visual representations and the rendering phase finally converts the visual primitives to pixel data, which can be fed to displays. In distributed environments, where each of these phases can be processed by a separate cluster (or a supercomputer), it is definitely economical to transport the output of the computation or the rendering phase to remote machines for display.

TeraVision can be envisioned as a hardware-assisted, network-enabled "PowerPoint" projector for distributing and displaying scientific visualizations generated on computers ranging from embedded systems to laptops to graphics clusters. A user who wants to stream visualization simply plugs the VGA or DVI output of the source computer into a TeraVision Box (also called a VBox) for transmitting it to displays across the network. Figure 1 below, depicts the system's capability of taking video inputs from a wide range of video sources and streaming them to a wide array of display technologies over high-speed networks.

In this paper we will first present the main issues pertinent to high-resolution video streaming and the related work done in the area. We will discuss the system architecture in some detail and some important problems that were addressed. In the following sections, results from tests done over local and wide area gigabit networks are presented. Finally, we will talk about SAGE (Scalable Adaptive Graphics Environment) and the OptIPuter [4][6] project as architectures for supporting data and display-rich collaborative visualization environments of the future. The main contributions of the paper are:

- Scaling and routing of streams generated by an array of MxN clusters to fit on a display driven by XxY displays.

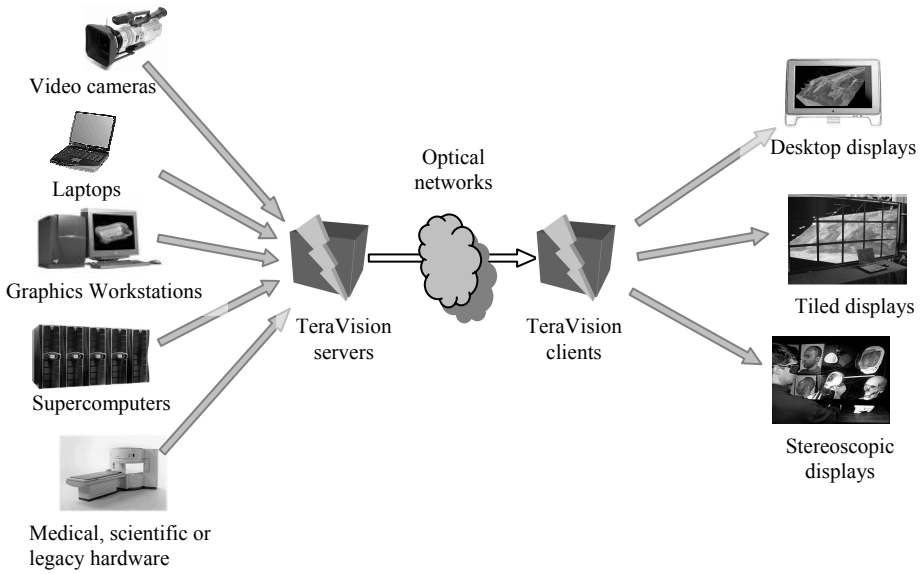

Figure 1. TeraVision servers can stream video from a variety of video sources and automatically resize the video stream to fit the client displays.

- Synchronization of the individual video streams that form a single larger stream.
- Streaming of high-resolution video over LFNs (Long Fat Networks).

2. Problems Associated with Video Streaming and Related Work

Video streaming for collaborative environments and scientific visualization applications has to be adaptable to a wide range of scenarios and faces many problems.

- *The video stream may be of the order of multiple gigabits per second*. Scientific visualization applications are capable of generating high-resolution video at high frame-rates. For example a single desktop with a resolution of 1280x1024 pixels at 24 bits per pixel and 30 frames per second translates to a network stream of 943 Mbps. Content-sensitive applications such as medical imaging cannot withstand artifacts generated from destructive compression and so in many cases the streams have to be transmitted raw. Though the system supports compression, we envision inexpensive dedicated network connections of multiple gigabits per second between visualization resources in the near future which are capable of transmitting real-time high-resolution uncompressed graphics. However, high bandwidth streaming puts considerable load on the sending and receiving machines to handle network traffic. A typical 32-bit Pentium Xeon machine at 1.8 GHz, utilizes 70% of its CPU to stream UDP data at 1 Gbps. Thus there is a need to offload the task of streaming video to dedicated machines.
- *The source and display machines have different resolutions and configuration*, e.g. laptop to tiled display or vice-versa. In this case, care has to be taken to appropriately scale the source video to the display at the viewing end.
- *The source hardware/software has to be designed to stream graphics*. Also the system responsible for converting the video data to network streams requires a network interface and should be aware of the networking protocols.
- *The source and display machines are different platforms.* Platform independence is a desired feature as distributed and collaborative environments are inherently heterogeneous in nature.
- *Ability to synchronize independent video streams.* When multiple video streams from complex display systems such as stereoscopic or tiled-display clusters need to be transmitted, there is a requirement to synchronize their capture at the video serving end and their display at the viewing end at every frame for the video to be displayed effectively [10].
- *The graphics may have to be streamed over LFNs.* With the introduction of inexpensive wide area optical networking solutions it is now possible to connect remote machines spread over a long distance using high-speed optical links. However, using these LFNs require specially tuned transport protocols, as conventional transport protocols were designed for slower networks [1]. Application writers are

typically unaware of networking details to handle this aspect.

Computing platforms typically provide some notion of remote desktop access. Tools such as VNC [11] or Microsoft's Remote Desktop [12] were designed to transmit screens of single desktops to remote computers over slow networks. They are designed to operate on event triggered streaming mechanisms which are not suitable for real-time streaming of scientific visualization or collaborative applications. Flexible scalable graphics systems such as Chromium [8] and Aura [9] are designed for distributing visualization to and from cluster driven tiled-displays. Applications have to be specifically written with the Aura API to enable them to transmit their graphics and Chromium was not designed to stream graphics over LFNs. Amongst hardware based approaches, Sandia National Laboratory's 'BeThereNow' [13] system provides excellent video quality and frame rates over LFNs using dedicated hardware. However it cannot be scaled to capture and stream from/to tiled or stereoscopic displays. IBM's Scalable Graphics Engine [17] is also a hardware based approach which allows for reception of pixels streamed over optical networks and can drive an array of displays synchronously. Currently it is limited to a collective bandwidth of 16 Gbps and does not support protocols for streaming over LFNs. Also because of the embedded nature of both these approaches, it is difficult for users to add their own video sources, compression modules or network protocols to the systems.

3. TeraVision: System Architecture

A TeraVision setup (Figure 2) typically consists of a server and a client connected over a gigabit network. The server has the video capture hardware for capturing high-resolution VGA or DVI inputs and the client can receive the streams and display them at various resolutions. The box captures the source signal at its native resolution, digitizes it and broadcasts it to other networked TeraVision boxes. Though the capture card is one of the preferred means of getting a video stream to the server, the video source can be any other video device such as a USB/FireWire camera or even the software generating the visualization itself.

In another configuration, using multiple VBoxes one can also transmit an entire tiled-display provided there are sufficient VBoxes at each end-point (Figure 3). Since the tile configuration and resolution can be different at either ends, the TeraVision boxes provide the scaling mechanism to automatically detect and 'fit' the source video to the destination display. Similarly two VBoxes can be connected to the twin-heads of a stereoscopic system to stream stereoscopic visualizations. The VBoxes take responsibility for the synchronization for simultaneous capture of concurrent video streams on the server side and the synchronization for displaying the streams on the client side.

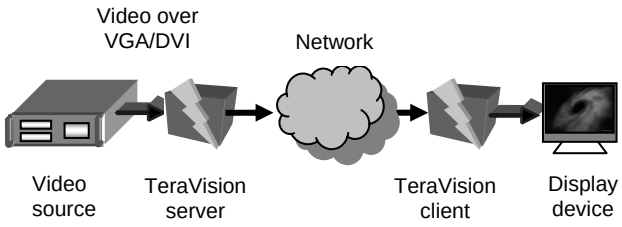

Figure 2. Typical TeraVision setup

Since the graphics output of the rendering source is typically transferred to a TeraVision server using a video cable, the source machine is not loaded for doing the network streaming. The TeraVision servers take up the load for compressing (if required) and streaming the video data. The source can be of any hardware/software platform and no changes have to be made to the source to perform the streaming. The system was designed to stream video over LFNs and contains TCP, UDP and IP Multicast modules which are tuned for this purpose. The system has also been designed to offer a framework for adding new video sources, compression modules or network protocols to the existing ones. Thus it is flexible enough to adapt to a wide range of scientific visualization and collaborative environments.

3.1 Concepts and Terminology

In this section, we will introduce the various components of the system and their relationships with each other.

TeraVision Server: The TeraVision server is the machine (or set of machines) which transmit the video streams. The direction of video is always from server to client. The server process is usually run on dedicated machines which have some video capture hardware. The video source can also be a software; for e.g. a set of pre-computed video frames being fed from the disk or RAM. One server process may be serving many clients at the same time and all the streams being sent out are synchronized with each other.

TeraVision Client: The clients display the incoming video streams on the machine's graphics output. The client does not require the capture hardware for its operation. It just needs a network interface fast enough to handle the video stream's bandwidth and a reasonably

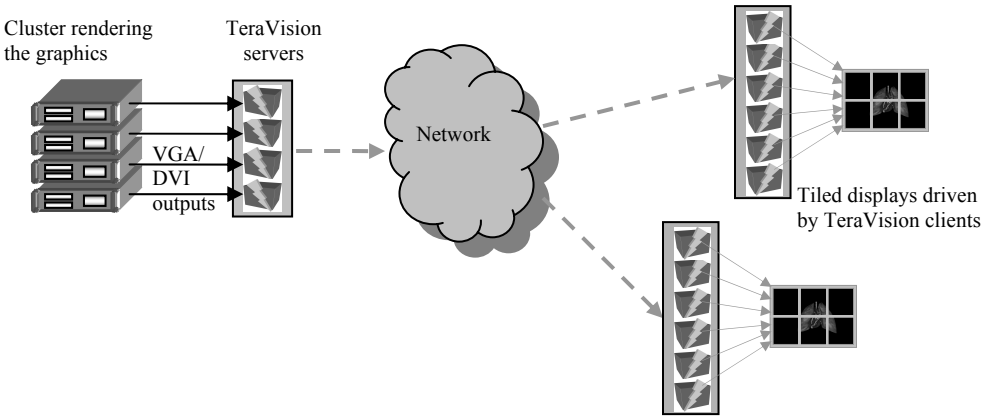

Figure 3. Example of streaming tiled displays using multiple VBoxes at each site.

Figure 4. A user streams her 1600x1200 laptop screen to Electronic Visualization Laboratory's 6400 x 3072 pixel tiled display at 10 fps.

Figure 5. Demonstration of three TeraVision servers streaming video to three different sections of the tiled-display. Each video stream is independent of each other but the screens for each stream are synchronized.

fast graphics card. One client may receive its video streams from one or more servers. All the component video streams that make up the display on a client are synchronized before being shown.

TeraVision Master: Each TeraVision server or client process either operates in the *Master* or the *Slave* mode. A TeraVision site would typically consist of one *master* and multiple *slaves*. Since there can be multiple servers (or clients), there has to be one point of control for all the VBoxes at a site and the *master* process serves that purpose. The *master* process is responsible for synchronizing all *slave* processes registered with it. The user interacts only with the *master* process for controlling all the VBoxes at a site. The *master* is also the one that sends the synchronization messages to all slaves.

TeraVision Slave: The *slave* processes are spawned by a *master* and depend on the *master* for providing the sync messages, user interaction messages and control plane information. Figure 6 graphically depicts the relationship between the *master* and the *slave* processes.

Sync Channel: The sync channel is a high-priority, low-latency channel used by peer server (or client) processes to closely synchronize the video streams across processes running on different machines. This is one of the more important components as it affects the throughput of the system to a large extent. Since a TeraVision server (or client) site may consist of many processes with each process sending or receiving many video streams, they have to be synchronized in real-time, so that the video at the display end does not appear to be non-uniform.

Control Channel: The control channel is an asynchronous message-passing channel that is hosted by all the TeraVision processes. The channel is implemented over TCP sockets and allows processes to pass control plane messages between each other.

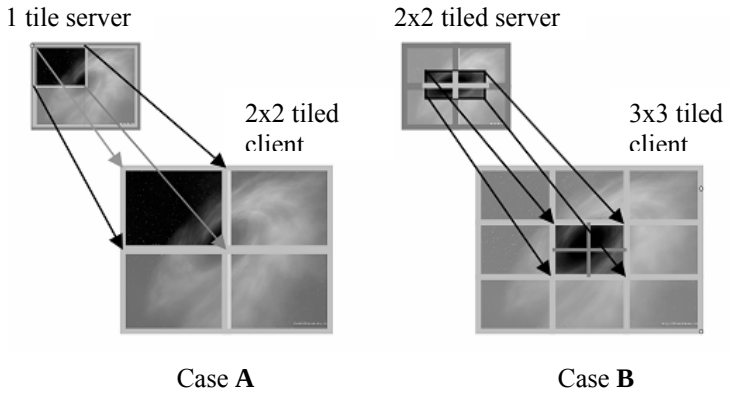

Figure 7. Case A depicts a single tile server sending its video to a 2x2 client. In this case the server has to service 4 clients and each client connects to one server. In the second case, the server is a 2x2 tile configuration and the client is a 3x3 tiled display. If we consider the center tile on the client, we see that it has to receive 4 video streams from 4 different servers.

4. Important System Issues

In this section we will talk about the main problems that were addressed for TeraVision.

4.1 M x N to X x Y scaling

In the case of streaming one tiled display to another, the configuration and resolution of the source tiles could be different from the client-side display. In such a scenario TeraVision will scale the source video to the client display. This can be taken as a general case and single displays or stereo displays can be considered as special cases of tiled displays.

During the initial handshake phase, all the slaves on the server side register with their masters. The master thus is aware of the total pixel resolution on the server side. Similarly on the client side, the master is aware of the collective resolution of the display. The master client then proceeds to calculate the video stream mappings for all the clients. Since there can be any number of tiles on either side, a server might be serving multiple clients at any given time or one client could be receiving its video montages (rectangular pieces making up the final video) from multiple server (Figure 7).

We make the assumption that the all tiles on either side have the same resolution. Consider the following parameters:

X_s - pixel width of each server tile.
Y_s - pixel height of each server tile.
X_c - pixel width of each client tile.
Y_c - pixel height of each client tile
M_s - Number of horizontal tiles on the server (numbered 0 to M_s-1)
N_s - Number of vertical tiles on the server (numbered 0 to N_s-1)
M_c - Number of horizontal tiles on the client (numbered 0 to M_c-1)
N_c - Number of vertical tiles on the client (numbered 0 to N_c-1)

The horizontal scale factor, $h_{sf} = (M_s * X_s) / (M_c * X_c)$
The vertical scale factor, $v_{sf} = (N_s * Y_s) / (N_c * Y_c)$

Then the remaining scaling parameters can be calculated from the following relations:

Number of horizontal client montages,

$$n_{hm} = \alpha_i \text{ for } i = 0,$$
$$= \alpha_i - \alpha_{i-1} + 1, \text{ for } i = 1 \text{ to } (M_c - 1)$$

Where $\alpha_i = (M_c/M_s) * i$, for $i = 0$ to $(M_c - 1)$

Similarly, number of vertical client montages

$$n_{vm} = \alpha_i \text{ for } i = 0,$$
$$= \alpha_i - \alpha_{i-1} + 1, \text{ for } i = 1 \text{ to } (N_c - 1)$$

Where $\alpha_i = (N_c/N_s) * i$, for $i = 0$ to $(N_c - 1)$

Now for every client montage that exists, there is a

corresponding server montage. In order to be able to retrieve and show the corresponding server montages, the clients need to find the following parameters,

O_{xs} – horizontal offset in a server tile
O_{ys} – vertical offset in a server tile
m_{xs} – montage width on the server side
m_{ys} – montage height on the server side

Knowing the number of client montages, we traverse from left to right and top to bottom pixel by pixel for all the server tiles. And every time we encounter either a server tile boundary or (the corresponding) client tile boundary, we note the offsets O_{xs} and O_{ys} in that server tile and the width (m_{xs}) and height (m_{ys}) of the server montage.

Thus, server side montage pixel width = $h_{sf} * m_{xs}$
Server side montage pixel height = $v_{sf} * m_{ys}$

Knowing the number of montages, the scaling factors, server side offsets and montage dimensions, the clients can calculate the client side offsets of each montage too. Each client is then able to connect to one or more servers and request a part (or whole) of the video stream based on these parameters. After the montages have been streamed to the client, hardware support in the graphics card is used to do the final stretching to fit the display screens.

4.2 Sync Channel

The current implementation of the sync channel is done as a two way handshake over TCP/IP. The sync packets have to be sent from the master to all the slave processes with minimal delay to get the best results. One way to ensure good latency between processes is to switch off the Nagle's algorithm in the TCP stack. This can usually be done by setting the TCP_NO_DELAY option at the TCP socket level [3] as explained in RFC 896 [7]. Another requirement of this channel is real-time priority. The upper bound on the transmission time of sync packets is greatly affected by factors such as system load, scheduler's time slice resolution, buffering delays etc. Real-time implementation of this channel will be carried out in the future.

4.3 Video Data over UDP

When video streams are sent over UDP and packet losses or packet duplication occurs, care has to be taken to recover from these errors. If the receiver fails to detect any anomaly, then the video frames are distorted. A protocol was built over UDP streams for this purpose. This protocol numbers every UDP packet of a video frame and allows the receiver to correctly place the packet in its right place in the video buffer despite losses, out-or-order arrival or duplication.

5. Tests and Observations

In this section we shall present results obtained with TeraVision systems over local and wide area gigabit networks. All machines used for the tests were equipped with gigabit Ethernet adapters and a 100 BaseT interface. The video data was sent over the gigabit interfaces and the synchronization messages are passed over the Fast Ethernet interface.

5.1 Specifications

The system has been implemented for Linux and Windows operating systems. For the Linux machines, we used Dual Intel Xeon CPUs at 1.8 GHz, 512 MB RAM running Red Hat 7.3. The video source was emulated in software. For the windows machines, we used Intel Xeon CPUs at 1.5 GHz, 512 MB RAM running Windows 2000 Professional. The video source was Foresight Imaging's I-RGB-200 video-capture card. All machines used 1 Gbps network interface cards. The LAN tests were done over a gigabit switch. Tests for LFNs were conducted between SARA, Amsterdam and EVL (Electronic Visualization Laboratory), Chicago over the Starlight network [14] and also between GRNET (Greek Research and Technology Network), Greece and EVL.

The video frames used for the tests are 1024x768 at 24bpp unless otherwise specified. All UDP tests were done with 1500 byte MTUs and TCP tests utilized 64 Kbytes flow windows. The prototype systems use capture cards that have drivers for the Windows OS only. Thus all results done with the Windows machines as the server imply the use of the capture hardware. The Linux servers used a video source emulated in software.

5.2 Experiments

The bandwidths shown are the maximum possible that could be achieved by the system under the given test conditions. The UDP bandwidths indicated in the results are the maximum sustained throughputs noted without any packet losses.

Performance over TCP and UDP

Tables 1 and 2 show the maximum throughput achieved with TCP and UDP streams over the LAN and WAN test-beds. We note that UDP performance for

Windows machines is limited to less than 200 Mbps because of the standard 1500 byte MTUs used. If we increase the size of the MTU, the throughput improves but the 1500 byte MTU was a standard test condition for all UDP streams used for the tests.

Effect of Synchronization and Scalability

Table 3 shows the effect of synchronization on system throughput. We observe that using one server as we increase the number of clients, the total sending bandwidth of the server decreases gradually but its CPU utilization increases. However since increasing the number of clients reduces the amount of video data received by each client, the CPU utilization on the clients goes down. Table 4 shows that with one client as we increase the number of servers, the amount of data being received by the client increases and its CPU usage goes up. This creates a bottleneck causing TCP's congestion mechanism to reduce the sending rate on the servers [3] and we see the CPU utilization of the servers decreasing.

Effect of different video frame sizes

In the following tests, shown in Figure 8 and 9, we vary the frame sizes. The tests shown here were done over TCP and UDP on LAN between one sending (Windows) machine and one receiving (Linux) machine. We notice that the throughput of the system is nearly constant even as the frame sizes are varied.

Table 1. System performance with TCP

Tests	Bandwidth used (Mbps)	Effective FPS*	Server CPU usage	Client CPU usage
LAN (Linux to Linux)	703.12	37.4	65.4 %	98 %
LAN (Windows to Linux)	534	28.4	60 %	64 %
Starlight (Linux to Linux)	110	5.8	15 %	9 %
GRNET (Windows to Linux)	110	5.8	54 %	10 %

Table 2. System performance with UDP

Tests	Bandwidth used (Mbps)	Effective FPS*	Server CPU usage	Client CPU usage
LAN (Linux to Linux)	930.6	49.5	64 %	94 %
LAN (Windows to Linux)	180	9.6	80 %	17 %
Starlight (Linux to Linux)	285	15.2	26 %	27 %
GRNET (Windows to Linux)	190	10.1	60 %	100%

Table 3. Effect of synchronization and scalability on clients

Number of Clients	Bandwidth used (Mbps)	Effective FPS*	Server CPU usage	Per Client CPU usage
1	534	28.4	60 %	64 %
4	530	28	95 %	17 %
9	503.84	26.8	97 %	7.5 %

Table 4. Effect of synchronization and scalability on servers

Number of Clients	Bandwidth used (Mbps)	Effective FPS*	Per Server CPU usage	Client CPU usage
1	623	33	70 %	80 %
4	180	9.5	17 %	94 %
9	120	6.3	15 %	99 %

**FPS: Frames per second.*

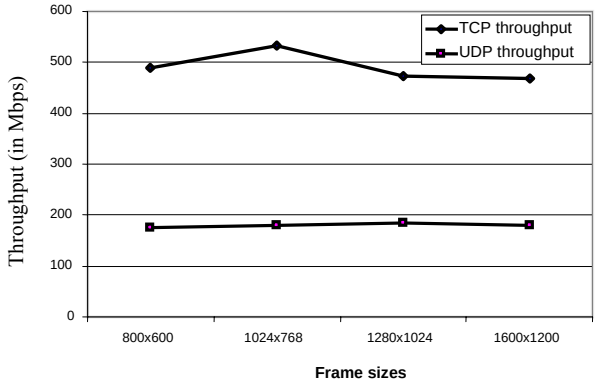

Figure 8. TeraVision throughput for different frame sizes.

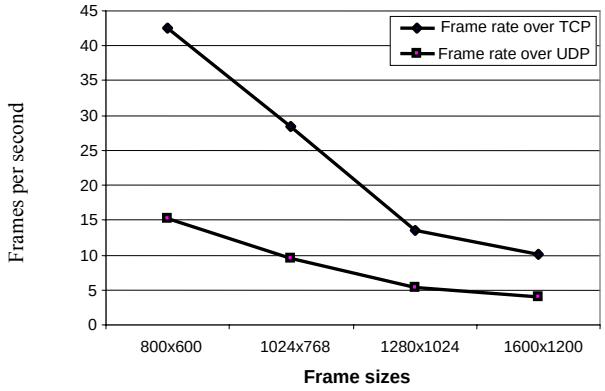

Figure 9. The frames per second decreases with increase in frame size

Multicast over gigabit networks

A WAN multicast test-bed was setup over gigabit links and high-bandwidth multicast tests were conducted between EVL, Technology Research Education and Commercialization Center (TRECC) and National Center for Supercomputing Applications (NCSA). The servers at EVL were used to transmit a captured laptop screen simultaneously to four separate clients at TRECC and one client at NCSA. Table 5 presents the results obtained from these tests. The bandwidth numbers are the maximum throughputs seen before packet losses were observed at either ends. In other words the receiving ends observed zero packet losses at the reported bandwidths.

All the receiving machines were running Linux. The Windows server used 1000 byte packets and the Linux server was configured with 1450 byte packets. It has been observed that the Windows OS shows better network throughput for UDP and Multicast with 1000 byte packets.

Table 5. System performance over multicast

Type of server	Bandwidth used (Mbps)	Effective FPS*	Server CPU usage	Client CPU usage
Windows	377	19.9	100 %	57 %
Linux	512	27.12	41 %	54 %

**FPS: Frames per second.*

7. Conclusions and Future Work

TeraVision provides a flexible and scalable solution for the display pushing problem. Users can stream the output of complex visualization systems, such as tiled-displays, over LFNs without making any modifications to the source software/hardware. The system scales the visualization, generated by the source machines, automatically to the client's display. The solution also distributes the load for streaming the high-resolution video streams to dedicated machines. This is a very important feature for computationally heavy visualization application environments. TeraVision provides a framework which allows easy adaptation to many visualization scenarios and it is hoped that it will prove to be a useful tool for collaborative environments and the scientific community.

The following issues need to be addressed as part of the future work.

- The system currently is capable of multicasting single video streams to single displays only. The problem of multicasting to and from tiled-displays requires more work and can be solved if it is assumed that the client configuration at each receiving site is the same.
- UDP may not provide the ideal solution in many scientific visualization scenarios because of the possibility of packet-losses and the resulting artifacts. We need a transport layer, which can provide the performance of UDP but with the reliability of TCP. EVL has been working on such a streaming protocol called the RBUDP [1]. In the future, TeraVision will provide a plug-in for RBUDP.
- As discussed in section 4.2, real-time guarantees are required from the OS to enable very good synchronization. We hope to incorporate real-time support in the future versions of TeraVision.

TeraVision is part of a larger research framework called the OptIPuter [4]. The OptIPuter is a National Science Foundation funded project to interconnect distributed storage, computing and visualization resources, using IP mechanisms over photonic networks. Recent years have seen a large increase in the capacities of optical networks, while the costs have plummeted. This allows one to experiment with a new paradigm in distributed computing - where dedicated optical networks serve as the computer's system bus; and compute clusters taken as a whole, serve as the peripherals in a widely distributed computing platform. The OptIPuter is aimed at solving research problems for the scientific community, such as in the areas of geo-science and bio-informatics [4][6].

The techniques developed in TeraVision are the basis for the OptIPuter's Scalable Adaptive Graphics Environment (SAGE) - an architecture for supporting cyber meeting rooms and laboratories which are "wall-papered" with ultra-high-resolution displays [16]. The impetus for the work on SAGE comes from the two main problem areas. The first is that applications written for one graphics environment have to be re-designed before they can be run under other environments. For example: visualization tools that are developed for desktop computers are rarely able to take advantage of the processing power of a cluster of graphics computers; conversely visualization tools developed for clusters rarely function on desktop computers. Secondly, the ability of visualization software and systems to scale in terms of the amount of data they can visualize, and the resolution of the desired visualization, is still an area of intensive computer graphics research [2][15]. SAGE addresses the need to support heterogeneity and scalability by decoupling graphics rendering from graphics display and taking advantage of affordable ultra-high-bandwidth networks to bridge them.

The SAGE architecture will allow multiple rendering nodes or clusters to access a virtual frame-buffer across the network. The framework will intelligently partition the graphics pipeline to distribute the load. Factors such as computing and rendering capabilities of the participating machines will decide the load distribution. Thus, unlike TeraVision where only pixels are routed to the display machines, SAGE would also route geometry and custom graphics formats. The framework will also support the notion of multiple collaborators simultaneously accessing a display space through a shared "window" manager.

Acknowledgements

We would like to thank Cees de Laat at University of Amsterdam for providing the endpoints at SARA in Amsterdam to perform experiments during the development of TeraVision. Alan Verlo, Lance Long, Pat Hallihan, Paul Wielinga and Hans Blom provided us with networking and system support. Also the valuable inputs by Naveen Krishnaprasad, Shalini Venkataraman, Javier Girado, Yong-Joo Cho and Greg Dawe are greatly appreciated. We also want to thank our collaborators for their help with conference demos; Fotis Karayannis from GRNET (Greek Research and Technology Network), Mike Papka from ANL (Argonne National Labs) and Dave Semeraro from NCSA.

This work was supported in part by the OptIPuter grant from the National Science Foundation-Cooperative Agreement ANI-0225642. It was also supported by the Office of Naval Research through an award from the Technology Research Education and Commercialization Center (TRECC).

References

[1] E. He, J. Leigh, O. Yu , T. A. DeFanti, "Reliable Blast UDP: Predictable High Performance Bulk Data Transfer", IEEE Cluster Computing 2002, Chicago, Illinois, Sept 2002.

[2] W. Blanke, C. Bajaj, D. Fussell, and X. Zhang, "The Metabuffer: a Scalable Multiresolution Multidisplay 3-D Graphics System using Commodity Rendering Engines." Tr2000-16, University of Texas at Austin, February 2000.

[3] W. R. Stevens, "Unix Networking Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI," Addison Wesley, 1998, pp.357.

[4] J. Leigh, L. Renambot, T.A. DeFanti , M.D. Brown , E. He, N.K. Krishnaprasad, J. Meerasa, A. Nayak, K. Park, R. Singh, S. Venkataraman, C. Zhang, D. Livingston, M. McLaughlin, "An Experimental OptIPuter Architecture for Data-Intensive Collaborative Visualization", 3rd Workshop on Advanced Collaborative Environments (in conjunction with the High Performance Distributed Computing Conference), Seattle, WA 06/22/2003 - June 2003

[5] J. Leigh, A. Johnson, K. Park, A. Nayak, R. Singh, V. Chowdhry, "Amplified Collaboration Environments", VizGrid Symposium, Tokyo, November 2002.

[6] T.A. DeFanti, J. Leigh, M.D. Brown, D.J. Sandin, O. Yu, C. Zhang, R. Singh, E. He, J. Alimohideen, N.K. Krishnaprasad, R. Grossman, M. Mazzucco, L. Smarr, M. Ellisman, P. Papadopoulos, A. Chien, J. Orcutt, "Teleimmersion and Visualization with the OptIPuter," Proc. of the 12th International

Conference on Artificial Reality and Telexistence (ICAT 2002), The University of Tokyo, Japan, December 3-6, 2002, to be published by Ohmsha/IOS Press.

[7] John Nagle. Rfc896: Congestion Control in IP/TCP Internetworks. Technical report, Internet Assigned Numbers Authority, Jon Postel, USC/ISI, 4676 Admiralty Way, Marina del Rey, DA 90292, 1984.

[8] G. Humphreys, M. Houston, Y. Ng, R. Frank, S. Ahern, P. Kirchner, and J. T. Klosowski, "Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters", Proc of SIGGRAPH 2002.

[9] D. Germans, H.J.W. Spoelder, L. Renambot, H. E. Bal, "VIRPI: A High-Level Toolkit for Interactive Scientific Visualization in Virtual Reality", Proc. Immersive Projection Technology/Eurographics Virtual Environments Workshop (IPT/EGVE), Stuttgart, Germany, May 2001.

[10] Y. Chen, H. Chen, D. W. Clark, Z. Liu, G. Wallace, K. Li., "Software environments for cluster-based display systems", First IEEE/ACM International Symposium on Cluster Computing and the Grid, May 2001.

[11] http://www.realvnc.com

[12] http://www.microsoft.com/windows2000/docs/rdpfandp.doc

[13] http://www.sandia.gov/newscenter/publications/sandiatechnology/2003/st2003v5no1.pdf

[14] http://www.startap.net

[15] G. Stoll, et. al., "Lightning-2: A High-Performance Display Subsystem for PC Clusters," Proceedings of Computer Graphics (SIGGRAPH 2000).

[16] http://www.evl.uic.edu/cavern/sage

[17] Parallel Graphics and Interactivity with the Scaleable Graphics Engine, Kenneth A. Perrine, Donald R. Jones, William R. Wiley, Proceedings of the 2001 ACM/IEEE conference on Supercomputing.