INTRODUCTION

In the last few years, scientists and researchers have given a great deal of attention to the area of remote visualization of scientific datasets within collaborative environments. This recent interest has been fueled by the use of interactive viewing as the primary means by which researchers explore large datasets. However, researchers often need to extend this interactiveness in order to collaborate with colleagues who are geographically separated. Most current remote visualization tools allow multiple parties to view images from different locations, but pose problems with efficiency and user interactivity.

![Diagram of the collaborative remote visualization system](image)

The remote visualization rendering pipeline with the client-server paradigm applied to it. As illustrated, the data is rendered (partially or fully) depending on the rendering method on the server and sent to the remote client via a variety of communication protocols. Our focus has been on improving the communication stage of the pipeline through the implementation of multICAST.

**Communication**

- **Server**: Unicast, Multicast
- **Compressor**: Server Manager
- **Renderer**: Rendering
- **Display**: User Interface

**User Interface**

- **Local UI**
- **Remote UI**

**Approach**

In general, popular recent approaches that address these shortcomings focus on improving two different areas of remote visualization:

1. Increasing network bandwidth utilization
2. Adjusting the amount of rendering performed on a local server versus a remote client in order to optimally utilize resources

In particular, researchers and developers use the client-server paradigm as a logical means to partition rendering responsibilities, efficiently utilizing valuable resources on both local and remote machines. In this project, we have built upon a prototype remote visualization application that applies this client-server paradigm, along with several rendering methods, in order to offer greater flexibility for remote viewing. As a part of the application framework, multiple clients may share remote visualization applications via a graphical user interface. We have implemented this tool as an extension of the SCRAN problem solving environment.

We have focused this project on the communication portion of the client-server rendering pipeline. Specifically, we have implemented multICAST of image data in order to improve network bandwidth utilization for an arbitrary number of clients that connect to the server. We have experimented with both reliable multICAST protocols as well as unreliable IP multICAST, comparing transfer rates with standard TCP point-to-point transfers in order to find the most efficient method.

**RESULTS**

For several reasons, we found reliable multICAST to be unsuitable for streaming real-time image data. First, we encountered excessive overhead with implementing reliability at the application layer. Second, since lost packets are usually unacknowledged, the retransmission of lost packets is generally not useful in real-time data transfer. Third, even with a reliable multICAST protocol that uses efficient NAK/ACK algorithms, receivers must wait for any packet in order to remain synchronized. Once we found that reliable multICAST could not give the transfer rate needed to compete with TCP point-to-point transfers of image frames, we chose to use IP multICAST. In our tests, IP multICAST yielded send times that are comparable to TCP send times. Packet loss was acceptably low given that we expect some loss in an interactive remote imaging application.

**RELATED RESEARCH**

The researchers at Lawrence Berkeley Labs have recently used UDP to transfer image data. However, their application, ViSual, provides an interesting contrast to our research because it also uses unreliable transfer of image data. In general, research indicates that in order to reach the network I/O levels needed for efficient image data transfer, unreliable transfer may offer a significant performance boost over reliable methods with an acceptable amount of packet loss.

**CONCLUSION**

We have integrated multICAST in order to achieve scalability, and improve the ability of multiple users to access a visualization application, allowing greater communication and flexibility with data exploration. We have also illustrated the collaborative potential of our remote visualization application through testing in a collaborative session between a standard Access Grid node and a Personal Interface Grid.

**FUTURE WORK**

- Research methods for dealing with packet loss for unreliable image data transfer
- Generate more data on the relationship of multICAST, compression, and various rendering methods
- Compare transfer times of other reliable multICAST protocols such as GM (Pretty Good MultICAST) and NEM (Reliable Reliable MultICAST)

**Table 1**: Comparison of effective network bandwidth for varying access grid nodes (2.5GHz P4, 128MB RAM, Windows XP)

<table>
<thead>
<tr>
<th>Access Grid Node 1</th>
<th>Access Grid Node 2</th>
<th>Access Grid Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Bandwidth</td>
<td>Network Bandwidth</td>
<td>Network Bandwidth</td>
</tr>
<tr>
<td>1.5Gbps</td>
<td>1.5Gbps</td>
<td>1.5Gbps</td>
</tr>
<tr>
<td>1.0Gbps</td>
<td>1.0Gbps</td>
<td>1.0Gbps</td>
</tr>
<tr>
<td>0.5Gbps</td>
<td>0.5Gbps</td>
<td>0.5Gbps</td>
</tr>
</tbody>
</table>

**Table 2**: Comparison of packet loss for varying data transfer rates (100MB, 250MB, 500MB)

<table>
<thead>
<tr>
<th>Data Transfer Rate</th>
<th>Packet Loss (access grid node 1)</th>
<th>Packet Loss (access grid node 2)</th>
<th>Packet Loss (access grid node 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100MB</td>
<td>0.1%</td>
<td>0.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td>250MB</td>
<td>0.2%</td>
<td>0.2%</td>
<td>0.2%</td>
</tr>
<tr>
<td>500MB</td>
<td>0.4%</td>
<td>0.4%</td>
<td>0.4%</td>
</tr>
</tbody>
</table>

**Network Bandwidth**

- **TCP (Point-to-Point)**
- **IP MultICAST**

**Data transfer rate**

**On the left is a diagram of a standard Access Grid node, very similar to the one at the SCI Institute. On the right is a diagram of a Personal Interface Grid, which is a compact version of an AG node with a multiple monitor display instead of a single monitor. The AG node and the PIG exchange audio and video data to enable remote collaboration. For further details, see our server on the PIG with a remote user on the Access Grid node.**

**This is a view of the server and local client being run on a Personal Interface Grid. This shows a direct view of images to both the local client as well as the remote client in the Access Grid Node. The video from the Access Grid Node can be seen on the right monitor.**

**Here is one of our Access Grid developers (Richard Coffey) giving a demonstration of the Access Grid to participants on an Access Grid node. The client illustrates remote viewing and control of a dataset being rendered by SCRAN on a personal interface grid. The video from the Personal Interface Grid can be seen on the display wall.**

**www.sci.utah.edu**

This work was supported by the NSF Partnership for an Advanced Computational Infrastructure. Research Experience for Undergraduates Program (IMSC - Innovative Approaches to Enhancing Performance Parallelism for Magnetic Fusion) which is funded under the DOE Office of Science NEXUS program. DE-FG02-01ER25140.