Hindawi Publishing Corporation
Advances in Multimedia
Volume 2008, Article ID 615186, 17 pages
doi:10.1155/2008/615186

Research Article

RICE: A Reliable and Efficient Remote Instrumentation Collaboration Environment

Prasad Calyam, Abdul Kalash, Ramya Gopalan, Sowmya Gopalan, and Ashok Krishnamurthy

Cyberinfrastructure and Software Development Group, Ohio Supercomputer Center, 1224 Kinnear Road, Columbus, OH 43212, USA

Correspondence should be addressed to Prasad Calyam, pcalyam@osc.edu

Received 9 March 2008; Accepted 3 November 2008

Recommended by Ghassan Alregib

Remote access of scientific instruments over the Internet (i.e., remote instrumentation) demand high-resolution (2D and 3D) video image transfers with simultaneous real-time mouse and keyboard controls. Consequently, user quality of experience (QoE) is highly sensitive to network bottlenecks. Further, improper user control while reacting to impaired video caused due to network bottlenecks could result in physical damages to the expensive instrument equipment. Hence, it is vital to understand the interplay between (a) user keyboard/mouse actions toward the instrument, and (b) corresponding network reactions for transfer of instrument video images toward the user. In this paper, we first present an analytical model for characterizing user and network interplay during remote instrumentation sessions in terms of demand and supply interplay principles of traditional economics. Next, we describe the trends of the model parameters using subjective and objective measurements obtained from QoE experiments. Thereafter, we describe our Remote Instrumentation Collaboration Environment (RICE) software that leverages our experiences from the user and network interplay studies, and has functionalities that facilitate reliable and efficient remote instrumentation such as (a) network health awareness to detect network bottleneck periods, and (b) collaboration tools for multiple participants to interact during research and training sessions.

1. INTRODUCTION

Increased access to high-speed networks has made remote access of computer-controlled scientific instruments such as microscopes, spectrometers, and telescopes widely-feasible over the Internet. Some of these instruments are extremely expensive and could be worth several hundred-thousand dollars. Hence, a major benefit of remote instrumentation is that it allows remote users to utilize these instruments when they are not in use by local users. In addition, routine maintenance and operation of the instruments require significant investment in staffing. Thus, instrument labs can charge remote access on an hourly usage basis to obtain a better return-on-investment on the instruments. Further, remote instrumentation avoids duplication of investment in instrument labs for funding agencies. In fact, the National Science Foundation is mandating remote instrumentation to be available with all their funded instruments [1]. Besides the above advantages, remote instrumentation fosters education and hands-on training of instruments as well as collaboration for remote users. The collaboration enables multiple remote researchers, each with unique expertise, to jointly analyze samples such as metals, proteins, and tissues. All of the above advantages, especially for training and collaboration, drastically shorten the development process involved in innovations related to materials modeling, biological specimens' analysis for cancer research, and so forth. At the same time, they improve user convenience and significantly reduce research and training costs.

Although there are several advantages, remote instrumentation is demanding in terms of network resource consumption. This is because remote instrumentation sessions involve high-resolution (2D and 3D) video image transfers with simultaneous real-time mouse and keyboard controls. If appropriate network bandwidth is not allocated, network congestion occurs that can impact user *quality of experience* (QoE). In addition, user QoE is affected by network fault events such as optical fiber cuts, route asymmetry, and route flapping that degrade network performance. The user

QoE affected by such network bottlenecks is measured by obtaining subjective opinions of user satisfaction after completion of a remote instrumentation session. The network bottleneck could cause impaired video images at the user, which in turn could lead to improper user control of the microscope's mechanical moving parts. Such improper user control may ultimately result in physical equipment damages that are prohibitively expensive to fix. Hence, it is vital to understand the interplay between (a) the user keyboard/mouse actions toward the instrument, and (b) the corresponding network reactions for transport of instrument video images toward the user, for reliably supporting remote instrumentation.

Assuming that a sample has been shipped to an instrument lab and has been loaded into an instrument, there are two basic use-cases of remote instrumentation. The first use-case is called *remote observation*, where a remote user or multiple remote users only view real-time (2D and 3D) instrument video images. The remote user(s) direct an operator physically present at the instrument to perform all the control actions over a telephone or VoIP call. The second use-case is called *remote operation*, where a remote user or multiple remote users view the instrument video images and also control the instrument in real-time. The first use-case is preferred in cases where the remote users are not familiar with the instrument functionalities. It is also preferred if the intermediate network path between the user and the instruments has bottlenecks. The second use-case is preferred for both local and remote users in cases where human presence around the sample could cause undesirable effects. For example in microscopy involving electron microscopes, human presence increases ambient temperature, which alters properties of materials being analyzed at subangstrom levels on the microscope. Nevertheless, both use-cases require collaboration tools that support voice communications (i.e., *VoIP*) and instant messaging (i.e., *chat*) for communicating efficiently during remote instrumentation sessions. For the multiuser case, collaboration tools are required to (i) show who is controlling/viewing the session (i.e., *presence*) and (ii) manage control privilege amongst the users (i.e., *control-lock passing*) such that at any given instant, only one user controls the instrument.

There are two major parts to this paper. In the first part, we study the complex interplay characteristics between the user and the network during remote instrumentation sessions. As an exemplar for the interplay characterization, we focus on the remote access of electron microscopes (i.e., remote microscopy). However, our work is equally relevant for other computer-controlled scientific instruments. We first present an analytical model for characterizing *user* and *network* interplay during remote microscopy sessions in terms of *demand* and *supply* interplay principles of economics, respectively. The various remote microscopy system states affected by transient network conditions are also modeled. To obtain the trends of the session model parameters, we set up a remote microscopy testbed in cooperation with The Ohio State University's Center for Accelerated Maturation of Materials (CAMM). On this testbed, we use a novel methodology to perform QoE experiments involving actual novice/expert users for a variety of network conditions in LAN/WAN connections. We also present the analysis of the subjective and objective measurements obtained from the experiments. Our analysis provides insights about how network health impacts user behavior and ultimately user QoE.

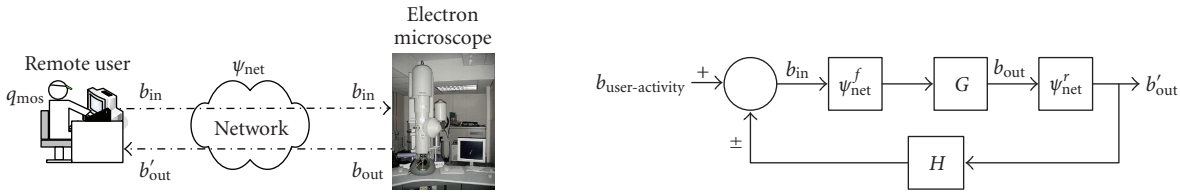
In the second part, we describe our *Remote Instrumentation Collaboration Environment* (RICE) software that leverages our user and network interplay study findings to (a) cope with network bottlenecks, and (b) cater to the multiuser requirements of remote observation and remote operation. In this context, we describe the RICE software functionalities that improve reliability and efficiency of multiuser remote instrumentation sessions that traditionally relied upon off-the-shelf *virtual network computing* (VNC) solutions [2]. The functionalities to improve reliability include real-time network health monitoring coupled with network performance anomaly detection using a "plateau-detector algorithm." This algorithm warns and blocks user's control actions during network congestion periods. We also describe our "session-signaling protocol" used in the RICE tools that improve efficiency of multiuser collaboration. The collaboration tools include VoIP, chat, presence, and control-lock passing that are absent in off-the-shelf VNC solutions. Finally, we present potential applications of RICE for research and training purposes that require multiuser remote instrumentation capabilities.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 presents the analytical model for user and network interplay characterization. Section 4 describes the remote microscopy testbed and results of the QoE experiments performed on the testbed. Section 5 describes the RICE software features and its applications for research and training. Section 6 concludes the paper.

2. RELATED WORK

There are several efforts in the United States that are aimed at serving the remote instrumentation needs of researchers and students. Gemini Observatory [3] is an initiative that uses Internet2 to allow remote users to manipulate their twin telescopes. NanoManipulator [4] is another initiative that uses Internet2 to allow remote control and visualization of images from their scanning probe microscopes. Similar remote instrumentation efforts are being supported in other countries also. A notable effort is being led by the National Institute of Materials Science in Japan, where remote instrumentation is being made available to the public and high school education programs [5]. As a part of this effort, remote observation of insects, plants, IC devices, and metals that have been preloaded in a remote-site's scanning electron microscope is being enabled at the National Museum of Emerging Science and Innovation in Tokyo.

One of the early works that developed novel applications for remote operation were done at the Massachusetts Institute of Technology [6]. They developed a custom software for remote control of a Zeiss microscope using a graphical interface running on a workstation computer. The software



(a) Session parameters

(b) Closed-loop control system representation

Figure 1: Basic remote microscopy system.

also allowed several remote users to simultaneously view the microscope in a conference inspection mode, enabling collaboration amongst remote users. Lawrence Berkeley National Laboratory also has developed a custom software application to control their Kratos 1500 keV microscope during in situ experiments [7]. The controls include adjusting external stimuli, adjustment of specimen position and orientation, and manipulation of microscope controls such as illumination, magnification, and focus. To cope with network bottlenecks, they developed schemes to locally automate stage control and microscope focus. Their application has been tested on the Internet along several paths including paths to Berkeley from Washington D.C. and Kansas City. Several other studies have also evaluated performance of remote instrumentation using custom software over the Internet. For example, Research Center for Ultra High Voltage Electron Microscopy (UHVEM) at Osaka University collaborated with National Center for Microscopy and Imaging Research (NCMIR) at University of California San Diego to conduct remote instrumentation experiments on their 3-million volt transmission electron microscope over intercontinental links [8]. The custom software developed by NCMIR has evolved over the years to keep up with the developments of networks, operating systems, and application development tools. The latest variants of their software feature platform-independent Java-based applications for remote instrumentation of several different instruments. These applications have also been integrated into web-services and middleware frameworks [9] that couple remote instrumentation with data and computation services.

Recently, several off-the-shelf remote access solutions have emerged that are either software-based or hardware-based. The most commonly used solution is the software-based virtual network computing (VNC) solution [2] that has several variants such as UltraVNC [10] and RealVNC [11]. It requires preinstalled software at both the instrument and user ends. Alternately, there are hardware-based VNC solutions that are also referred to as *Keyboard, Video and Mouse over IP* (KVMoIP) solutions developed by vendors such as ThinkLogical [12] and Avocent [13]. These solutions use custom hardware and require a pair of encoder and decoder appliances to be installed at the instrument and user ends. Recently, hybrid VNC solutions have also been developed by vendors such as Adder [14] that requires a hardware appliance at the instrument end, and a software client at the user end. Several instrument labs use such off-the-shelf solutions. For example, Oak Ridge National Laboratory uses off-the-shelf VNC solutions for remote control of their High Flux isotope Reactor [15]. Similarly, the California State Polytechnic University also uses off-the-shelf VNC solutions in their Ocean Engineering Program [16].

VNC solutions use raw or copy-rectangle or JPEG/MPEG encoding for video image transfers and TCP for keyboard and mouse control traffic. For sending the video image transfers, VNC uses a Remote Frame Buffer (RFB) protocol that supports various pixel formats such as ZRLE, Zlib, Raw, and Hextile. The pixel updates using the RFB protocol are demand-driven because pixel updates are sent (a) to respond to an explicit TCP-based request from a client, and (b) to update the client's display when there are changes at the server's display. The compression latency of VNC is dependent on factors such as the network health, as well as the client/server CPU speed, other-application task loads, and video card capabilities.

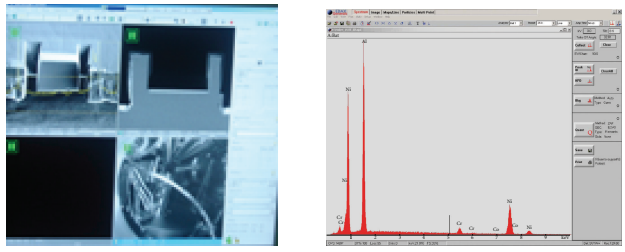
Given the free availability of software-based VNC solutions, QoE evaluations for these solutions can be extensively found in the remote instrumentation literature. However, to the best of our knowledge, there is no literature on systematic QoE evaluations for KVMoIP-based remote instrumentation. The QoE evaluation results presented in Section 4 of this paper focus on the KVMoIP VNC solution on both LAN and WAN paths. We also believe that our work is the first to present an analytical model and characterize user and network interplay in remote instrumentation sessions. Our RICE software presented in Section 5 is based on the UltraVNC solution but has several enhancements targeted for network-aware and collaborative remote instrumentation sessions. Our work in this paper is part of The Ohio State University's CAMM VIM program [17]. This program uses Ohio Supercomputer Center's (OSC) regional network (i.e., OSCnet) to allow remote industry such as Timken and defense labs such as AFRL to access their collection of the world's most powerful scanning/transmission electron microscopes.

3. REMOTE MICROSCOPY SESSION MODEL

In this section, we first describe the parameters involved in a typical remote microscopy session. Next, we model their interactions in different system states borrowing the supply and demand terminology from economics.

3.1. System description

Figures 1(a) and 1(b) show a basic remote microscopy system and its closed-loop control system representation with



(a) Image with live video feeds (b) Image with text and graphs

Figure 2: Comparison of video activity levels in instrument image transfers.

the different session parameters, respectively. The remote user physically controls the functions of the microscope by interacting with a graphical user interface (GUI) application using keystrokes and mouse moves/clicks via VNC or KVMoIP (console). Examples of microscope functions include adjusting stage position, lens focus, and magnification levels. The GUI application actually resides on a computer directly connected to the microscope's video output and control input ports. Figures 2(a) and 2(b) show two distinct video activity levels (i.e., temporal and spatial characteristics in the GUI application images sent from the microscope to the remote user). We can notice that the images contain live video feeds of instrument cameras with high video activity levels, or text and graphs with low video activity levels.

Let b_{action} be the average bit rate of the TCP control traffic that is generated due to keystrokes and mouse moves/clicks at the user end to accomplish a particular microscope function. The user-activity input to the system during a session involving n microscope functions can be denoted by $b_{\text{user-activity}}$ given in

$$b_{\text{user-activity}} = \sum_{i=1}^{n} (b_{i\text{th action}}). \tag{1}$$

For such an input, the average video image transfer rate (i.e., RTP media traffic output at the microscope end (b_{out})) can be denoted as follows:

$$b_{\text{out}} = \psi_{\text{net}}^{f} G(b_{\text{in}} + b_{\text{seed}}), \tag{2}$$

where ψ_{net}^{f} is the network connection quality between the user and the microscope. The network connection quality refers to the end-to-end throughput that is affected by network congestion and network fault events. The G corresponds to the input-output scaling factor which is unique for a microscope function. The b_{seed} corresponds to the rate at which periodic intracoded frames (I-frames) are sent from the encoder (at the microscope) to the decoder (at the user) for quick image refresh upon recovery from network partition events during a session.

Although b_{out} is sent from the microscope, there are two network factors that could degrade the average video image transfer rate at the user end (b'_{out}). The first factor is the network connection quality of the reverse path (i.e., between the microscope and the user (ψ_{net}^{r})). The second factor is the available bandwidth in the intermediate network path. As shown in the following equation, if adequate available bandwidth is provisioned, b'_{out} will be equal to b_{out}; otherwise, b'_{out} is limited to b_{net}, which refers to the bottleneck hop bandwidth:

$$\begin{aligned} b_{\text{net}} &= \min_{i=1..\text{hops}} b_{i\text{th hop}}, \\ b'_{\text{out}} &= \min(b_{\text{snd}}, b_{\text{net}}). \end{aligned} \tag{3}$$

The degradation of b'_{out} manifests to users as video signal impairments such as frame freezing, blurriness, and tiling [18]. Based on the positive or negative b'_{out} feedback received at the user end from the microscope, the subsequent user behavior determines the session state. We refer to this system-state control parameter that is dependent on the user behavior as H. Details of how H parameter impacts the different system states are described in Section 3.2. We can thus express b_{in} as follows:

$$b_{\text{in}} = b_{\text{user-activity}} - H b'_{\text{out}}. \tag{4}$$

Using substitutions in (1)–(4), we can derive the closed-loop transfer function in the classical form as shown in the following equation; this function fully describes the order, type, and frequency response for a remote microscopy system:

$$\frac{b'_{\text{out}}}{b_{\text{user-activity}}} = \frac{G\psi_{\text{net}}^{f}\psi_{\text{net}}^{r}}{1 \pm G\psi_{\text{net}}^{f}\psi_{\text{net}}^{r}H}. \tag{5}$$

Ultimately at the end of a session, the overall user QoE (q_{mos}) will depend on both the effort a user had to expend to perform n actions (i.e., $b_{\text{user-activity}}$) and the perceivable video image quality (i.e., b'_{out} during those actions). Hence, q_{mos} can be expressed as follows:

$$q_{\text{mos}} = f\left(\underbrace{b_{\text{user-activity}}}_{\text{Demand}}, \underbrace{b'_{\text{out}}}_{\text{Supply}}\right). \tag{6}$$

From (6), we can make an analogous comparison of $b_{\text{user-activity}}$ and b'_{out} to the "demand" and "supply" terminology used in economics, respectively. In traditional economics, an increase in demand levels for a commodity causes an increase in supply levels of the commodity. This in turn increases the demand, as the increased supply in large numbers generally drives down the overall commodity price. As long as both the demand and supply increase hand-in-hand by deriving reinforcement from each other, the economy (analogous to q_{mos}) is considered to be in a productive state. However, this is not always the case in the demand and supply reinforcement effect seen in remote microscopy with respect to q_{mos}. The overall network health in both the forward and reverse paths (ψ_{net}) adds complexity in the relationship of the demand and supply variables as elaborated in the next subsection, which severely affects the q_{mos}.

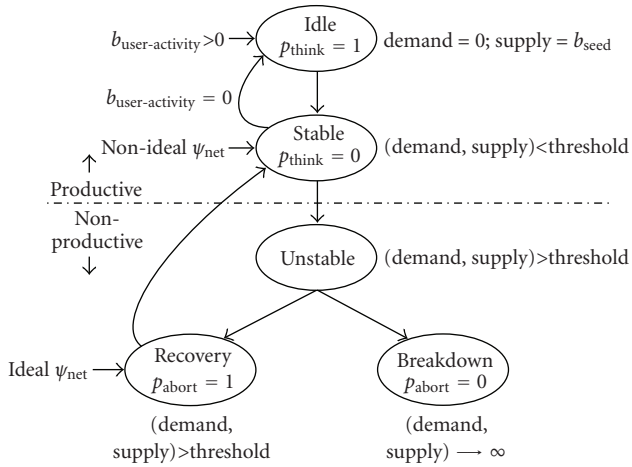


Figure 3: Remote microscopy system state transitions.

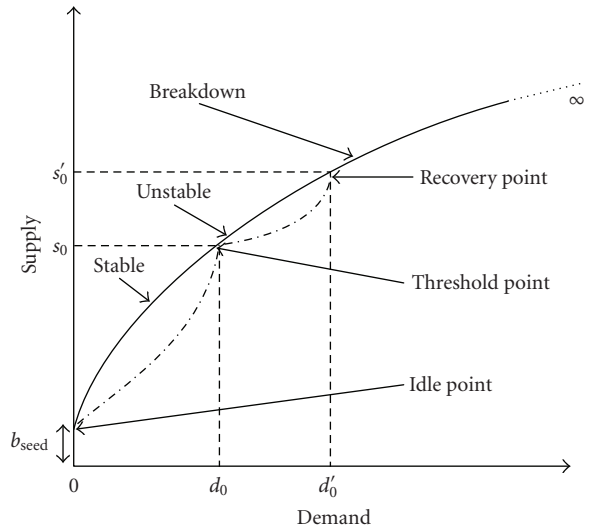


Figure 4: Remote microscopy system performance at different supply and demand conditions.

As a note, the above remote microscopy session model can be applied for both remote observation as well as remote operation use-cases. Recall that the remote operation use-case employs inband TCP control traffic toward the microscope, whereas remote observation use-case employs an out-of-band voice channel (e.g., a telephone) for directing control messages to a local user at the microscope. If we assume that a reliable voice channel exists between the two users and that the local user is responsive enough that the remote user does not perceive annoying control delays, the model is identical for both the use-cases.

3.2. System states

We now explain the interactions of the remote microscopy session parameters due to user behavior that affect the H parameter. The changes in the H parameter influence the $\pm$ sign (positive or negative feedback) of the denominator in (5) which in turn causes the different system state transitions shown in Figure 3. Initially, the system is in the "Idle" state when the user is inactive with a probability p_{idle} and the microscope GUI application is operational. In the Idle state, the demand is zero and the supply equals b_{seed} as shown in Figure 4. The remote microscopy session begins upon user-activity, and the demand and supply steadily increase. Assuming ideal ψ_{net} conditions at a given time t, the system attains a "Stable" state where the demand and supply are below the system's optimum performance threshold point (s_0, d_0). In this state, the user is successfully controlling the microscope functions and is being productive. We can now say that H is causing negative feedback in the system. At random times in this state, it is possible that a user will still be in session but idle in terms of control, presumably due to a thought process driven by a visual inspection of a sample's area of interest. Such an inactive user behavior brings the system back to its Idle state where the system is still productive. During such user inactivity times under ideal ψ_{net} conditions, we refer to p_{idle} as p_{think} as follows:

$$p_{\text{idle}}(t) = \begin{cases} p_{\text{think}}, & \text{if ideal } \psi_{\text{net}}(t), \\ p_{\text{abort}}, & \text{if nonideal } \psi_{\text{net}}(t). \end{cases} \tag{7}$$

If the ψ_{net} were to change to nonideal conditions due to network bottlenecks caused by network congestion and network fault events, the system would enter an "Unstable" state. Here, the demand and supply rapidly increase beyond the system's optimum performance threshold point. This is because the user in this system state experiences QoE degradation effects (e.g., frame freeze) that force him to misjudge his control actions that result in *unwanted* supply. This is subsequently followed by a retry of the previous actions before the unwanted supply transfer completes, which further increases the demand and the QoE degradation effects and so on. Soon, the system becomes nonresponsive to the increasing demand, and is pressured into handling large volume of unwanted supply that is introduced from the microscope end. It is important to note that although the demand and supply rapidly increase hand-in-hand beyond the threshold point, the system is nonproductive. We can now say that H is causing positive feedback in the system. If the user persists in his retry demand behavior, the system soon advances to a "Breakdown" state where the demand and supply tend to ∞. However, if the user aborts any actions and becomes idle at a recovery point (s_0', d_0'), the system transitions into a "Recovery" state. During such user inactivity times under nonideal ψ_{net} conditions, we refer to p_{idle} as p_{abort} as shown in (7). During the Recovery state, the demand and supply gradually tend toward the system's optimum performance threshold point. Once the ψ_{net} returns to ideal conditions (e.g., due to reduced network congestion or stabilization of the impulsive demand and unwanted supply), the system regains its Stable state and becomes productive again.

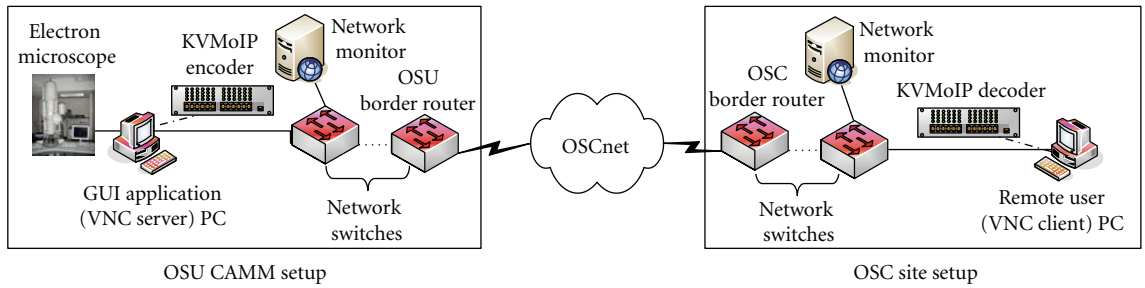


Figure 5: Remote microscopy testbed setup.

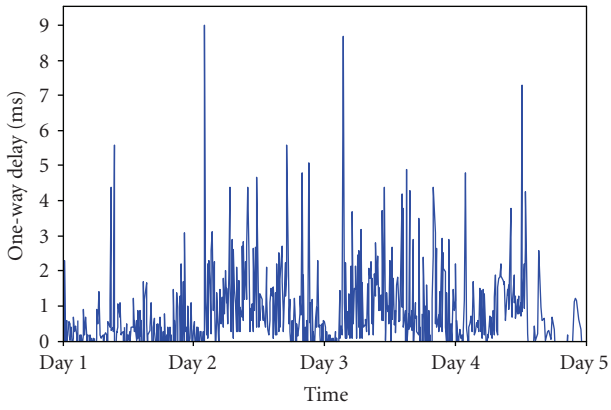


Figure 6: One-way delay between OSC and OSU CAMM.

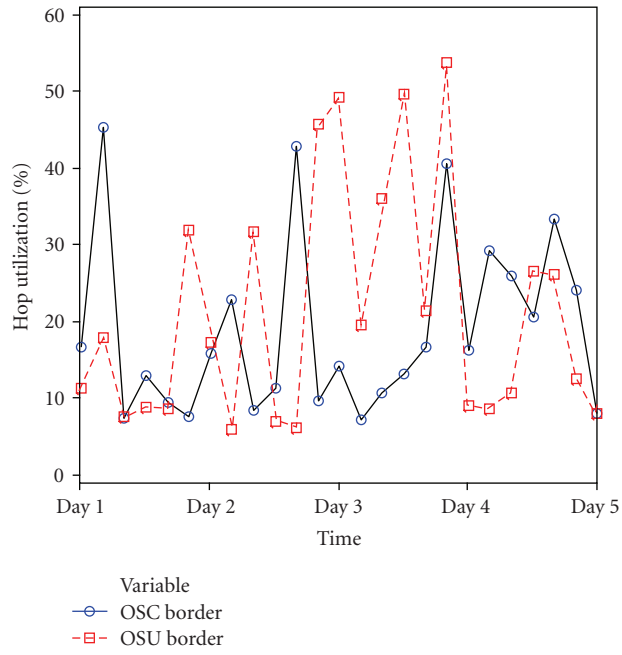


Figure 7: Border hop utilization at OSC and OSU.

4. REMOTE MICROSCOPY TESTBED EXPERIMENTS

In this section, we first describe the remote microscopy testbed used to obtain trends of the different session model parameters under different network conditions. Next, we explain the test cases and performance measurements collected during the QoE experiments on the testbed. Lastly, we discuss the QoE experiment results.

4.1. Testbed setup

For setting up the remote microscopy testbed, we collaborated with The Ohio State University's Center for Accelerated Maturation of Materials (OSU CAMM). The testbed featured four different network connections between the remote user console and the GUI application PC: (i) Direct GigE (ii) Isolated LAN (iii) Public LAN, and (iv) WAN. The Direct GigE connection had a Cisco GigE switch connecting the GUI application PC and the remote user console, which were in adjacent rooms. This connection represents the setup for avoiding users to be physically present at the microscope, especially when human presence around a sample is undesirable as explained in Section 1. The Isolated LAN connection was setup by including the CAMM's Cisco Catalyst 2924 switch to the Direct GigE connection. This connection represents remote microscopy for users in the same LAN as the microscopes, but in different lab rooms. The Public LAN connection was setup by including three additional Cisco Catalyst 2924 switches located at neighboring buildings to the Isolated LAN connection. This connection represents remote microscopy for users working from different LANs and different lab rooms. Finally, the WAN connection was setup as shown in Figure 5 via OSCnet between OSU CAMM and OSC. This connection represents remote microscopy for users at remote sites on the Internet.

The Direct GigE, Isolated LAN, and Public LAN connections were 100 Mbps switched full-duplex connections. To know the baseline performance of the 100 Mbps WAN connection, a number of measurements were collected over a 5-day period using the OSC-developed ActiveMon software [19]. The measurements indicated the available bandwidth, delay, jitter, and loss trends. Figure 6 shows the one-way delay measurements in the path between OSC and OSU (i.e., remote user to microscope) as measured by the OWAMP tool [20] in the ActiveMon measurement toolkit. We can observe that the one-way delay measurements are generally within 10 microseconds. Figure 7 shows the bandwidth utilization levels (sampled once every 4 hours) at the bottleneck hops

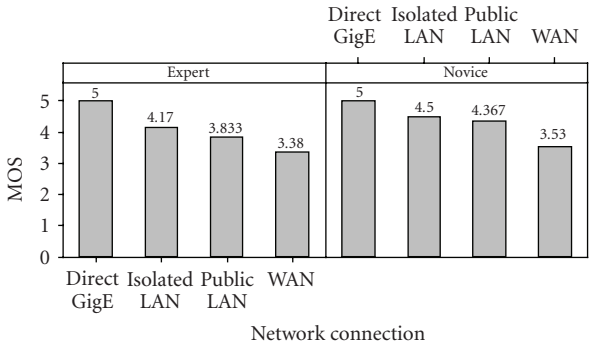


Figure 8: MOS (q_{mos}) rankings comparison for varying ψ_{net} conditions.

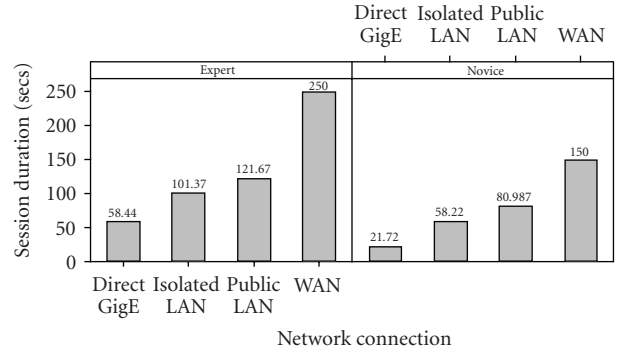


Figure 9: Session duration (T) comparison for varying ψ_{net} conditions.

(i.e., the border routers at OSU and OSC). We can observe that worst-case utilization level is ≈50% and thus, in general, there is at least about 50 Mbps available bandwidth in the WAN path. The other routers in the WAN path are the OSCnet routers with utilization levels <20%, typical to most backbone routers.

4.2. Test cases and measurements

The test cases involved performing preassigned tasks by actual users in remote microscopy sessions using a KVMoIP VNC solution [12] over the different network connections. Raw pixel format with copy-rectangle video encoding was used in all the test cases. The actual users (i.e., human subjects) were classified under two groups: (i) "novice" and (ii) "expert," with three human subjects in each group. The novice users performed a set of sequential tasks with simplistic actions: *Task-1:* move view from one location on the surface of sample material to another location, *Task-2:* focus on high-resolution imaging, and *Task-3:* change the quad-screen to a single screen and grab a high resolution image. The expert users performed a set of sequential tasks with advanced actions that require relatively more effort and skill: *Task-1:* eucentric height adjustment—stage movement in the Z-direction, *Task-2:* beam modulation—column alignment for best image, and *Task-3:* focus for high-resolution imaging.

During execution of the test cases, both objective and subjective measurements were collected. The objective measurements correspond to passive measurements of the control traffic (b_{in}) and video traffic (b'_{out}) collected using the popular TCPdump packet sniffing tool. The subjective measurements are the user QoE measurements, which are collected using the popular mean opinion score (MOS) ranking technique [21]. In this technique, at the end of each test case (i.e., remote instrumentation session task), the user is asked to rate his/her perceived QoE (q_{mos}) on a subjective scale of 1–5, with [1, 3) range being *Poor* grade, [3, 4) range being *Acceptable* grade, and [4, 5] range being *Good* grade. In addition to the MOS rankings, completion times (T) of novice and expert sessions were also recorded.

4.3. QoE experiment results

4.3.1. Network connection and user QoE

First, we analyze the impact of network connection quality on the user QoE MOS (i.e., q_{mos}) in a remote microscopy session. Figure 8 shows the average q_{mos} comparison between the novice and expert users for varying ψ_{net} conditions, with ψ_{net} being the highest for Direct GigE, and the lowest for WAN. For both types of users, we can observe that the q_{mos} rankings decrease notably with decreasing ψ_{net}. The q_{mos} equal 5 rankings of the novice and expert users while using the direct GigE connection indicates "at-the-microscope" QoE. Expectedly, for the other network connections, we can see that novice rankings are relatively more liberal than expert rankings due to the inherent intensity of the actions involved. The q_{mos} in the case of Isolated LAN and Public LAN are comparable. Further, the q_{mos} rankings for the WAN connection are in the acceptable grade, suggesting that user QoE in remote microscopy is highly sensitive to network congestion.

The average time to complete a set of predetermined user actions (i.e., session duration T) is another useful metric that provides insight about the user QoE. Figure 9 shows the session durations of the novice and expert for varying ψ_{net} conditions. For both types of users, we can observe that the session duration T increases with the decrease in ψ_{net}. For instance, the session duration T more than doubles in the Isolated and Public LAN connections in comparison to the Direct GigE connection. In the same context, we can see that the MOS ranking dip is higher for the expert (from 5 to 4.17) than the novice (from 5 to 4.5). We can generalize this observation of lesser MOS rankings for higher session durations across the other cases of both expert and novice users by comparing the session durations in Figure 9 with the MOS rankings in Figure 8. The longer session durations can be attributed to the additional effort (e.g., mouse moves/clicks, keyboard strokes, waiting for image transfer) involved while coping with network health fluctuations. Given that the difficulty level of expert user tasks is higher than novice user tasks, we can see that MOS ranking dip is higher in the expert user cases than the novice user cases.

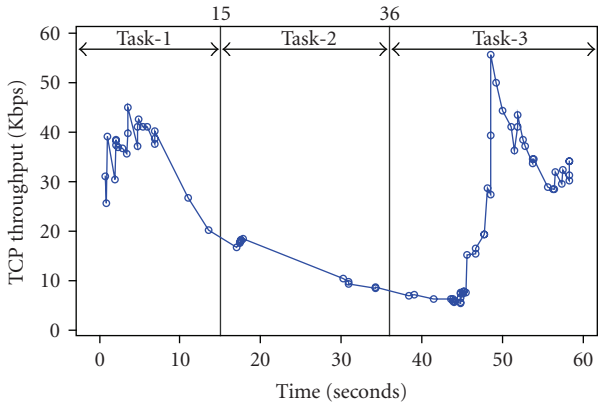


Figure 10: Control traffic (b_{in}) during an expert session on direct GigE network connection.

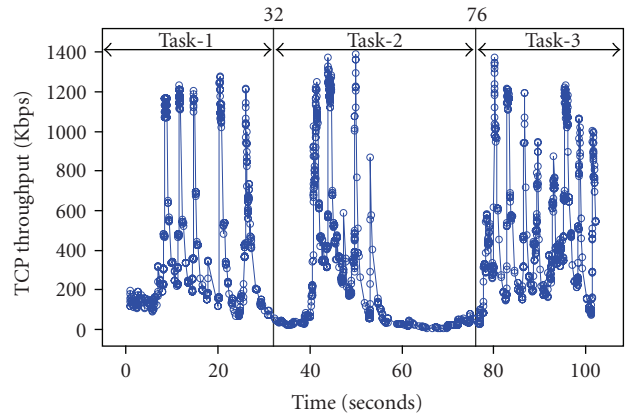


Figure 11: Control traffic (b_{in}) during an expert session on public LAN network connection.

4.3.2. Network connection and user control

Next, we characterize how the network connection quality impacts the trends of user control behavior (i.e., b_{in}). Figures 10 and 11 show the instantaneous b_{in} throughput levels during an expert session on Direct GigE and Public LAN connections, respectively. The throughput levels clearly show the amount of user effort required for accomplishing each of the three tasks of the session. Another notable observation is that user effort is considerably less ($\lceil b_{in} \rceil \approx 60$ Kbps) in the case of the Direct GigE connection as compared to the user effort ($\lceil b_{in} \rceil \approx 1400$ Kbps) on the Public LAN connection. Also, the throughput trends are significantly less dense in case of the Direct GigE connection as compared to the Public LAN connection. Due to space constraints, we do not show the throughputs for the WAN network connection, where the expert user effort was the most when compared to the other connections ($\lceil b_{in} \rceil \approx 2000$ Kbps).

We note that such an inverse relationship between the network connection quality and user control effort is a driver for the "congestion begets more congestion" phenomenon, where a user expends more effort (i.e., mouse moves/clicks and keyboard strokes) on poor network connections, which cumulatively adds to the congestion already inherent in the poor network connections. The nature of the "Unstable" and "Breakdown" states and their transitions explained in Section 3.2 can be attributed to the occurrence of this particular phenomenon with different intensity levels. The intensity levels are based on the instantaneous network connection quality and the impulsive user reactions to video signal impairments such as frame freezing.

4.3.3. User behavior and video image transfers

Lastly, we analyze how a user's control behavior and network conditions impact the video image transfers from the microscope at the user end (i.e., b'_{out}). Figure 12 shows the b'_{out} comparison between the novice and expert for varying ψ_{net} conditions. Cross-referring to the q_{mos} equals 5 results shown in Figure 8 for the direct GigE network connection, we can observe that obtaining an "at-the-microscope" user

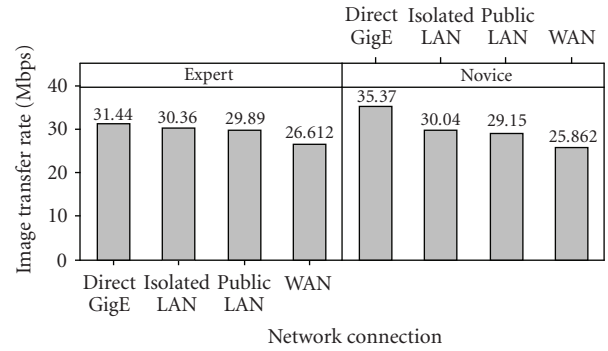


Figure 12: Image transfer rate (b'_{out}) comparison for varying ψ_{net} conditions.

QoE requires end-to-end available bandwidth in excess of 30 Mbps between the user and the microscope ends.

However, it is important to note that the average image transfer rates in remote microscopy can vary based on the microscope functions and user activity. Thus, they may not always be in the range of 30 Mbps. The reason for the high bandwidth consumption in our experiments can be attributed to the KVMoIP solution nature, and activity level in the experiments that had the quad-video panel images in the GUI application shown in Figure 2(a). Such a nature of video may not be present in every user session. For example, there may be sessions whose activity level may be similar to that of Figure 2(b), where the user is mainly plotting graphs, editing parameters while analyzing a sample. For such a session, the end-to-end available bandwidth requirement to achieve "at-the-microscope" user QoE will be considerably less.

5. REMOTE INSTRUMENTATION COLLABORATION ENVIRONMENT (RICE)

In this section, we describe the Remote Instrumentation Collaboration Environment (RICE) software application we have developed. The RICE design leverages our user and network interplay studies to effectively support the remote

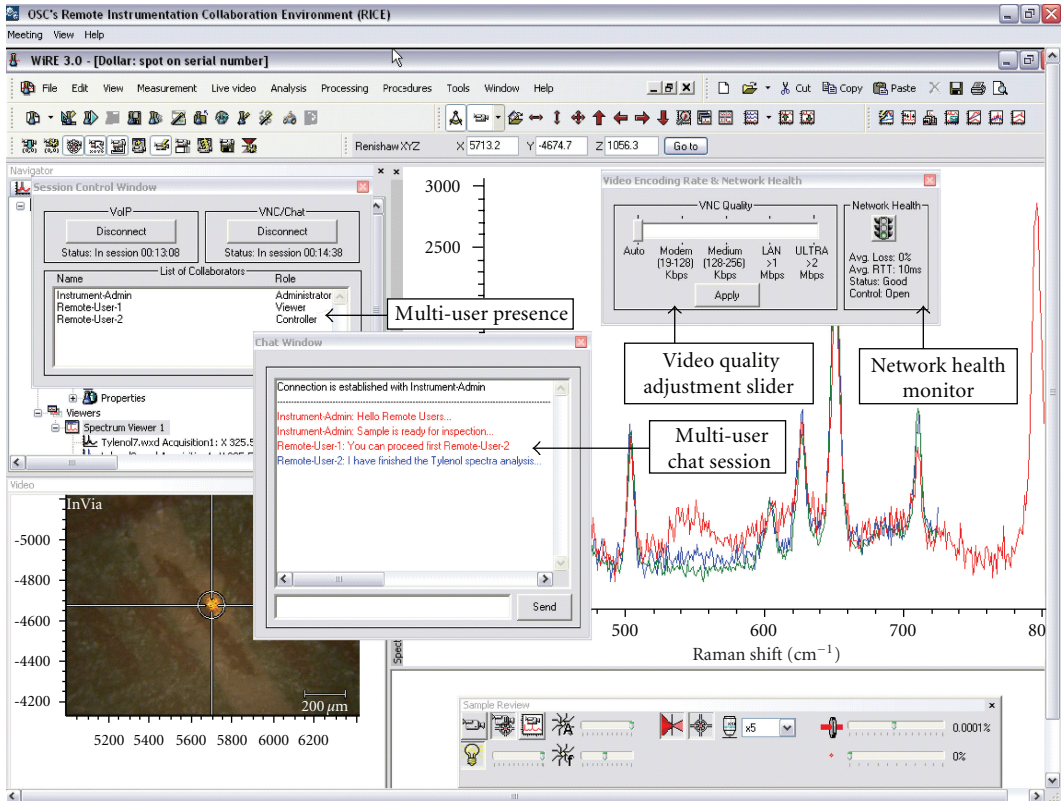


Figure 13: RICE in an active session.

observation and operation use-cases for instructors and researchers to train students and/or conduct research from remote locations on the Internet. RICE is based on the UltraVNC solution [10] but has several enhancements targeted for network-aware and collaborative remote instrumentation sessions that are reliable and efficient. The enhancements allow tuning of image feeds based on last-mile network bandwidth, and limit the provision of excessive control given by off-the-shelf VNC solutions during network congestion periods. Also, the enhancements provide collaboration tools (VoIP, chat, presence, control-lock passing) to orchestrate instrument-control amongst multiple remote users during remote operation of expensive and potentially dangerous instruments. Note that the network-aware control blocking and collaboration tools in RICE are features not available in off-the-shelf VNC solutions. The modular software design used in RICE permits customization to cater to unique considerations and requirements of a variety of users and instruments. Thus, RICE is a self-contained collaboration environment for multiple users to reliably and efficiently participate in remote instrumentation sessions.

Figure 13 shows the main functionality of RICE in an active session during remote operation of a Renishaw Raman Spectroscope deployed at the Department of Chemistry, The Ohio State University. The "video quality adjustment slider" is used by a RICE client user to manually adjust frame rates and video encoding rates based on the last-mile network bandwidth between the remote user site and instrument lab. A "network health monitor" shows real-time network health in terms of average round-trip delay (RTT) and loss. A traffic light indicates the network health grade as *Good* (green), *Acceptable* (amber), and *Poor* (red). The *Good* grade corresponds to RTT values in the range [0–150) milliseconds and loss values in the range [0–0.5)%; *Acceptable* grade corresponds to RTT values in the range (150–300) milliseconds and loss values in the range (0.5–1.5)%; *Poor* grade corresponds to RTT values in the range (>300] milliseconds and loss values in the range (>1.5]%. Such grade levels of RTT and loss have been obtained by studies such as [22, 23] that have conducted empirical experiments on the Internet for real-time multimedia applications. The network health monitoring in RICE is coupled with network performance anomaly detection using a "plateau-detector algorithm" that warns and blocks user's control-actions during impending and extreme network congestion periods, respectively. The control blocking is essential in spite of the fact that "limit switches" in some well-designed instruments mitigate damage due to user error. This is because it is practically infeasible to take into account all the possible user error cases when designing limit switches. The control status in RICE can be either "open," "warning," or "blocked" depending on the network congestion levels. Details of the network health monitoring coupled with the plateau-detector algorithm implementation in RICE are described in Section 5.1.

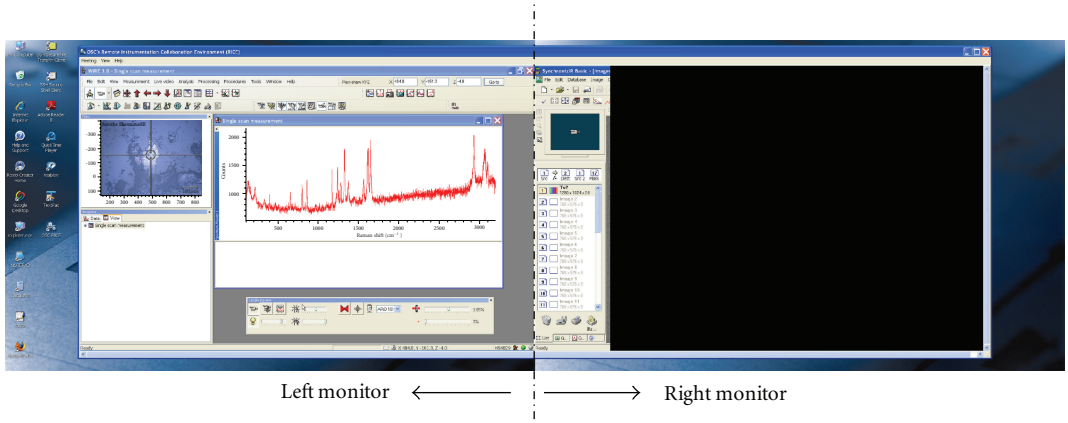


Figure 14: Problem with remote dual-screen resolution with UltraVNC.

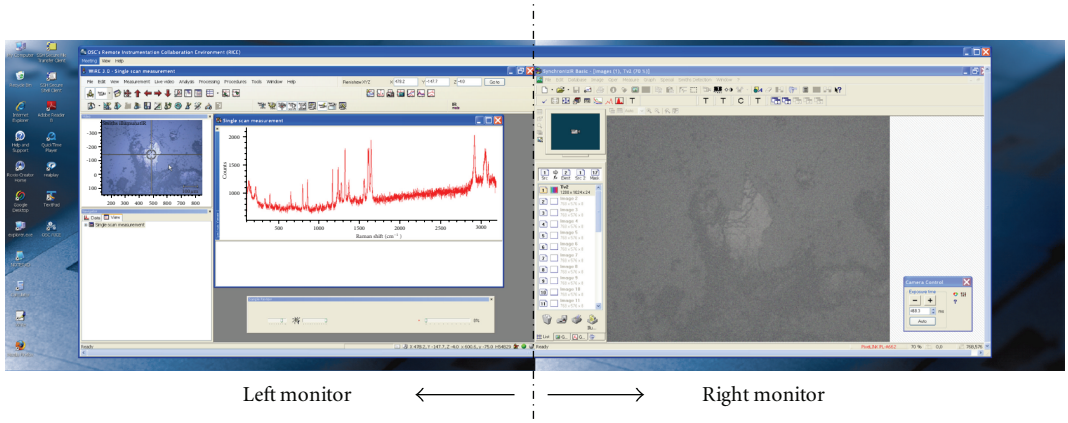


Figure 15: Increased remote dual-screen resolution with RICE.

RICE is designed to allow multiple remote users to simultaneously connect using RICE clients to the instrument console, which runs an instance of the RICE server. In a multiuser remote instrumentation session, RICE provides VoIP and chat functionality for collaboration as shown in Figure 13. The VoIP functionality is supported via the open-source OpenMCU software [24] that is integrated within RICE. The chat is supported by a custom chat application integrated within RICE that features colored text that can distinguish messages of the local user from the remote users. The names and roles of the multiple-users connected in a session are displayed and updated in real-time via a presence functionality. The roles correspond to the "administrator," "viewer," or "controller" (i.e., the users who possess the instrument control-lock, and the users who are observers). By default, the administrator always has control privileges, and he/she has the ability to pass another session control-lock to any one of the remote users. Once a remote user has a control-lock, he/she in turn can directly pass that control-lock in a session to any other remote user via the RICE client interface, without intervention of the administrator. All of the above multiuser collaboration tools of VoIP, chat, presence, and control-lock passing that are provided in RICE have been implemented using a "session-signaling protocol," whose details are described in Section 5.2.

RICE has a feature for simultaneously connecting to and transparently switching between two PCs for accomplishing inspection and analysis tasks of a remote instrumentation session. For example, in remote microscopy, a remote user will want to simultaneously connect and switch views between two PCs. One PC corresponds to the electron microscope console and the other PC corresponds to an energy dispersive spectroscopy (EDS) analysis PC. As another example, in remote telescopy, a remote user will want to simultaneously connect to a PC that shows the focal-plane instrument controls and another PC that is performing environment monitoring.

Further, we also developed a feature in RICE for handling remote dual-screen resolution. This feature allows a remote user to mimic an extended desktop setup with dual-monitors at the instrument computer. The extended desktop provides additional real-estate for users to run multiple application programs simultaneously. The openly-available UltraVNC distribution does not support dual-screen resolution at the remote VNC client as shown in Figure 14. We developed a software patch for the UltraVNC in RICE that increases the display resolution geometry to

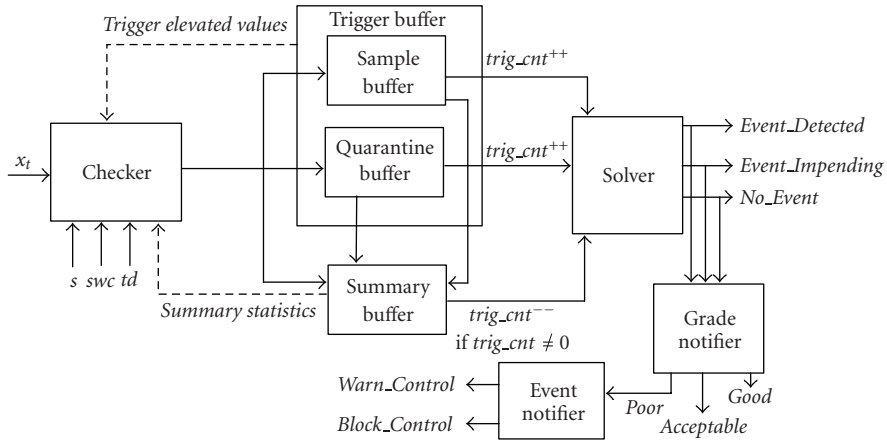


Figure 16: Plateau-detector block diagram.

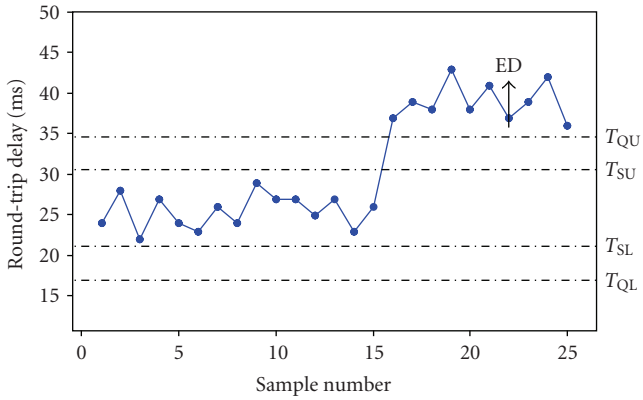


Figure 17: Plateau-detector thresholds illustration.

successfully render the dual-screen resolution at the remote VNC client as shown in Figure 15.

In the following subsections, we first describe the plateau-detector algorithm implementation and its integration in network health monitoring in RICE. Thereafter, we describe the session signaling protocol implementation to provide the collaboration tools functionality in RICE.

5.1. Network health monitoring

We use the popular *Ping* tool to monitor the network health in a RICE session in terms of average RTT and loss measurements. These RTT and loss metrics are useful to assess the impact of network health on the image transfer performance and interaction responsiveness as perceived by the user. Using RTT and loss measurements of Ping can generally indicate network congestion scenarios in most cases. However, it is possible in some cases that Ping's RTT measurements could misrepresent the network health status. For example, some routers are configured to handle Ping's ICMP packets differently to mitigate denial of service (DoS) attacks. In such cases, the RTT measurements might indicate higher values than actual, which can be construed as due to network congestion. Although there are several other sophisticated network measurement tools such as Iperf [25] and Pathchar [26] that can accurately measure available/per-hop bandwidth, they are not suitable for online network-aware adaptation in RICE. The reason is that these tools consume significant amounts of network bandwidth and CPU resources, and thus interfere with the RICE application performance. Besides, these sophisticated network measurement tools have been designed to be primarily used on network backbones for troubleshooting purposes rather than adaptation purposes in actual applications.

Our sampling rate of RTT and loss between the RICE client and server is periodic with a period of 6 seconds between consecutive samples. Each sample uses the default Ping settings (i.e., four ICMP packets each with packet size of 32 bytes). This results in time-series of instantaneous measurements of RTT and loss. We use x_t to denote an instantaneous measurement of RTT or loss that is input to a plateau-detector algorithm that detects network performance anomalies in real time. The purpose of the plateau-detector algorithm is to minimize the false-positive and false-negative anomaly alerts or triggers that arise if a naive mean-based method is used. A false-positive trigger is one that gets reported when there is no actual anomaly, whereas a false-negative trigger is one that does not get reported when in fact there is an actual anomaly. Two instances of plateau-detector algorithm run in-band in every RICE session, one to detect RTT anomalies, and the other to detect loss anomalies. The overall network health grade g and control status c are determined using an *OR* function of the two plateau-detector algorithm instance outputs.

Our plateau-detector algorithm implementation is similar to the basic implementations found in [27, 28] that are used for routine network monitoring. However, our implementation has several modifications to suit the time scales and user reaction times during network congestion periods in RICE sessions. In the mean-based method,

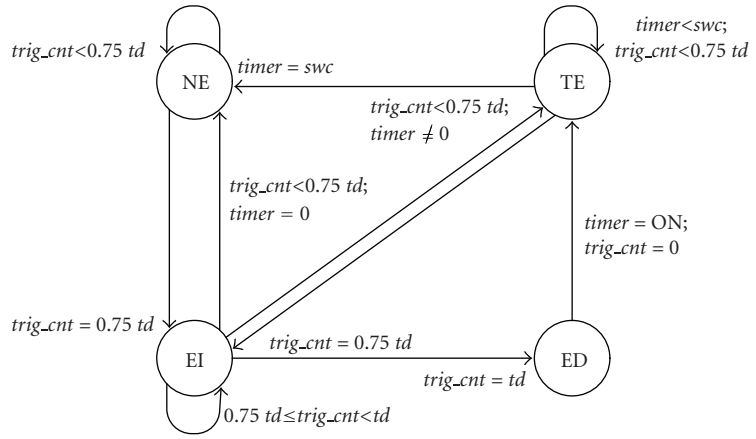


Figure 18: Plateau detector states and state-transitions.

the network health norm is determined by calculating the mean μ and comparing it with the standard deviation σ for a set of x_t values sampled most recently into a summary buffer. The number of samples in the summary buffer is user-defined and is specified using a summary window count *swc*. In comparison to the mean-based method, the plateau-detector requires two additional user-defined inputs called trigger duration *td* and sensitivity *s*. The trigger duration *td* specifies the duration of the anomaly before a trigger is signaled. Thus, the smaller the *td*, the faster a trigger will be signaled in the event of an anomaly. However, the *td* must be chosen to be large enough such that the transient spikes or bursts (i.e., noise events in network health are not signaled). The sensitivity *s* specifies the magnitude of the change for it to be considered as an anomaly. The choice of the *s* again requires consideration of the tradeoffs (i.e., a small *s* results in triggers for slight variations in network performance magnitudes, whereas a large *s* could overlook actual anomalies that should be detected).

Given our network health sampling rate of 6 seconds in RICE, we choose the *swc* to be equal to 20 so that anomalies can start getting detected within 2 minutes of initiation of a RICE session. In accordance with the *td* and *s* values selection in [27, 28], we choose *td* to be approximately 1/3rd *swc* (i.e., 7), and *s* value is chosen to be equal to 2. In this context, we remark that we have validated our selection of $s = 2$ using extensive simulations on synthetic and actual network health time series. The validation results are beyond the scope of this paper. Our general observation has been that values of *s* in the range of 2 produce the least number of false triggers (sum of false positives and false negatives), for a wide selection of *swc* and *td* values.

Figure 16 shows the different components of our plateau-detector algorithm implementation. The values of x_t are first input to a "checker" which compares whether the most recent x_t value lies within the upper and lower thresholds of the (i) summary buffer *sumbuff* (i.e., T_{SU} and T_{SL}) or (ii) quaratine buffer *qbuff* (i.e., T_{QU} and T_{QL}). These thresholds are illustrated in Figure 17 and are calculated using mean μ and standard deviation σ of the summary window as follows:

$$\begin{aligned} T_{SU} &= \mu + s * \sigma, \\ T_{QU} &= \mu + 2 * s * \sigma, \\ T_{SL} &= \mu - s * \sigma, \\ T_{QL} &= \mu - 2 * s * \sigma. \end{aligned} \tag{8}$$

If x_t values lie within these thresholds, the plateau-detector will be in the no event (NE) state shown in Figure 18. In this state, x_t values are put into the *sumbuff*. If x_t values go below T_{QL} or exceed T_{QU}, they are put into the quarantine buffer *qbuff*. Similarly, if x_t values cross T_{SL} and T_{SU}, they are put into the sample buffer *sampbuff*. If x_t is put into either *qbuff* or *sampbuff*, trigger count *trig_cnt* is incremented. Whereas, if x_t is put into *sumbuff*, *trig_cnt* is decremented as long as *trig_cnt* is nonzero. If *trig_cnt* exceeds $0.75^* td$ due to increasing number of x_t values going into *qbuff* or *sampbuff*, then the plateau-detector enters into an event impending (EI) state. If the *trig_cnt* drops below $0.75^* td$, then the plateau-detector returns to NE state. Otherwise, the plateau-detector stays in the EI state until *trig_cnt* equals *td*, after which it enters into an event detected (ED) state. Figure 17 shows an event detection occuring after x_t crosses the thresholds for the *td* of 7 samples. At this point, the *trig_cnt* is reset, and a *timer* is turned ON. The plateau-detector now goes into a trigger elevated (TE) state, where the upper and lower thresholds are calculated as follows:

$$\begin{aligned} T'_{SU} &= 1.2 * \max(x_t) \text{ in } \textit{trigbuff}, \\ T'_{QU} &= 1.4 * \max(x_t) \text{ in } \textit{trigbuff}, \\ T'_{SL} &= 0.8 * \max(x_t) \text{ in } \textit{trigbuff}, \\ T'_{QL} &= 0.6 * \max(x_t) \text{ in } \textit{trigbuff}. \end{aligned} \tag{9}$$

Until the *timer* equals *swc*, the elevated thresholds are used for comparing x_t. The reason for the trigger elevation is to avoid reporting of repeated triggers for the already detected anomaly. It is relevant to note that the plateau-detector can transition from TE state to EI state if another

```
(1) Input: summary window count (swc), sensitivity (s), trigger duration (td), instantaneous measurement (x_t)
(2) Output: mean μ, network health grade g, control status c
(3) begin procedure
(4) repeat
(5)   for each new x_t do
(6)       g = No_Event; c = Allow_Control
(7)       Calculate mean (μ) and standard deviation (σ) of summary window buffer (sumbuff)
(8)       if (timer == 0 /* NE state */) then
(9)         Calculate upper (T_SU, T_QU) and lower (T_SL, T_QL) thresholds using μ and σ of sumbuff and s
(10)      else
(11)        /* TE state */ Calculate upper (T_SU, T_QU) and lower (T_SL, T_QL) elevated thresholds using trigbuff
(12)      end if
(13)      /* Compare x_t with the thresholds and assign x_t to appropriate buffer */
(14)      if (T_QL > x_t or x_t > T_QU) then
(15)        Put x_t into quarantine buffer qbuff; Increment trig_cnt
(16)      end if
(17)      if (T_SL > x_t or x_t > T_SU) then
(18)        Put x_t into sample buffer sampbuff; Increment trig_cnt
(19)      end if
(20)      /* Report anomaly event types */
(21)      if (trig_cnt > td) then
(22)        event_type = Event_Detected; Copy sampbuff and qbuff to sumbuff; Reset trig_cnt
(23)      else if (trig_cnt > 0.75*td) then
(24)        event_type = Event_Impending
(25)      else
(26)        event_type = No_Event
(27)      end if
(28)      /* Update the buffers with latest network health norm */
(29)      if (trig_cnt == 0 and trigbuff not empty) then
(30)        Copy sampbuff to sumbuff; Empty qbuff
(31)      end if
(32)      /* Report μ of sumbuff, g and c*/
(33)      if (magnitude of x_t in Good grade level) then
(34)        return μ, g = Good, c = open
(35)      else if (magnitude of x_t in Acceptable grade level) then
(36)        return μ, g = Acceptable, c = open
(37)      else if (magnitude of x_t in Poor grade level) then
(38)        if (event_type == Event_Detected) then
(39)          return μ, g = Poor, c = Block_Control
(40)      end if
(41)      if (event_type == Event_Impending) then
(42)          return μ, g = Poor, c = Warn_Control
(43)        end if
(44)      end if
(45)    end for
(46) until end of RICE session
(47) end procedure
```

Algorithm 1: Plateau-detector algorithm implementation in RICE.

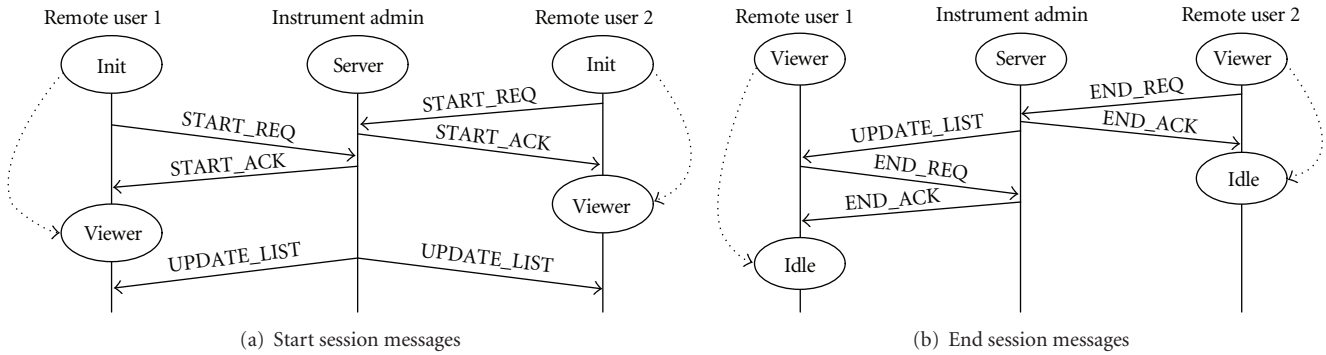
anomaly occurs due to x_t crossing the elevated thresholds. Once *timer* equals *swc*, and x_t does not cross the elevated thresholds, the plateau-detector returns to the NE state.

From Figure 16, we can see that the "solver" tracks the plateau-detector state transitions and outputs the *No_Event*, *Event_Impending*, and *Event_Detected* signals to the "grade notifier." If the network health grade is determined to be *Poor* as described in the RICE overview portion in Section 5, an "event notifier" is invoked to either warn the user of impending network congestion or to block any user control actions if a network congestion event is detected.

The implementation of the above plateau-detector algorithm in RICE can be formally described as shown in Algorithm 1 listing.

5.2. Collaboration tools

Herein, we describe the session signaling protocol we developed to provide the collaboration tools functionality in RICE. The protocol involves exchanging TCP-based messages on port 6000 between the RICE clients of the remote users and the RICE server at the instrument console.



(a) Start session messages (b) End session messages

Figure 19: RICE's Session Signaling Protocol messages for session initiation and termination.

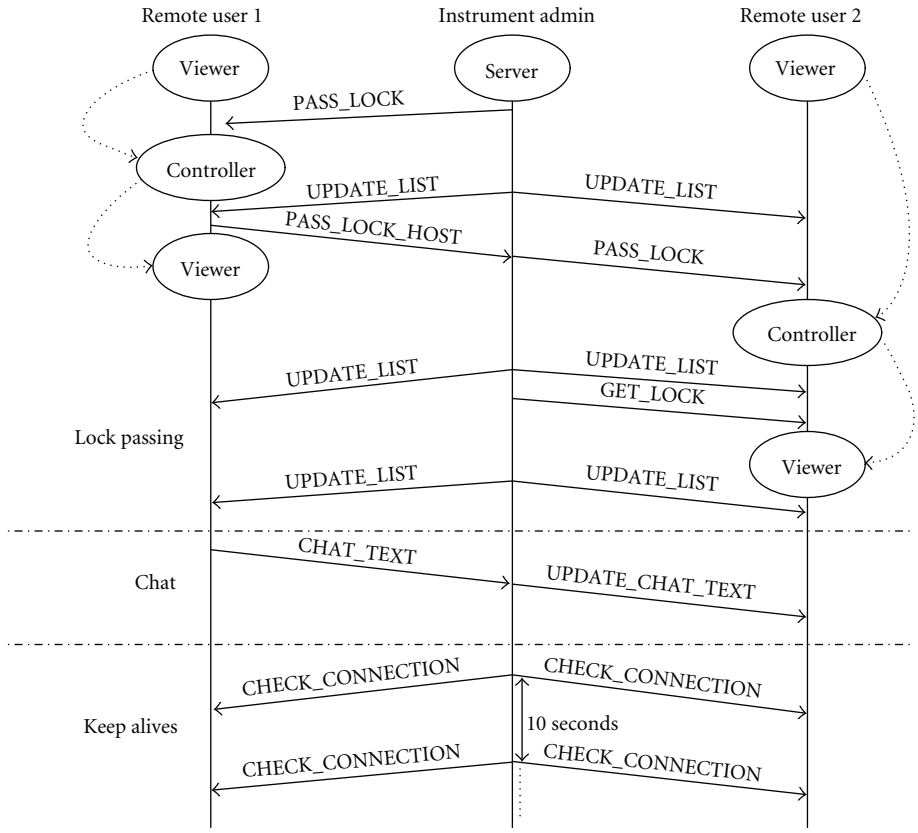


Figure 20: RICE's Session Signaling Protocol in-session messages.

Figures 19(a) and 19(b) show the start session and end session messages, respectively, for a RICE session involving two remote users. To start a session, a RICE client sends a START_REQ message to the RICE server. If a RICE server accepts the connection, it sends a START_ACK message back to the RICE client. Similarly, if a RICE client wants to terminate a session, it sends an END_REQ message to the RICE server, which in turn sends an END_ACK message to the RICE client. The UPDATE_LIST message maintains the latest presence information of all the users. It is event-driven and is sent from the RICE server to each of the RICE clients whenever there is a change of status in the RICE clients (i.e., whenever RICE clients join/leave the session, and whenever RICE clients switch between "viewer" and "controller" roles).

Figure 20 shows the different messages exchanged during a session when the collaboration tools are being used by the remote users. For the control-lock passing, we require the instrument administrator to initially pass the control-lock to any remote user. For this, the PASS_LOCK message is used. As we noted earlier, the instrument administrator always has a copy of the control-lock, and can retrieve the other control-lock from a remote user at any time. For the control-lock retrieval, the GET_LOCK message is used. Once

Table 1: List of RICE's session signaling protocol messages.

Message name	Purpose	Format
START_REQ	Sent from RICE client to RICE server to request a connection establishment	⟨START_REQ, USER_NAME, HOST_IP⟩
START_ACK	Sent from RICE server to RICE client to accept a connection request	⟨START_ACK, ADMIN_NAME⟩
UPDATE_LIST	Sent from RICE server to all connected RICE clients to update changes in user names/status	⟨UPDATE_LIST, USERS_STATUS[][]⟩
PASS_LOCK	Sent from RICE server to a RICE client to grant instrument control privileges	⟨PASS_LOCK⟩
PASS_LOCK_HOST	Sent from a RICE client to RICE server to pass control privileges to another RICE client	⟨PASS_LOCK, HOST_IP⟩
GET_LOCK	Sent from RICE server to a RICE client to revoke instrument control privileges	⟨GET_LOCK⟩
CHAT_TEXT	Sent from a RICE client to RICE server to send a chat text message to other connected RICE clients	⟨CHAT_TEXT, TEXT_INFO⟩
UPDATE_CHAT_TEXT	Sent from RICE server to all connected RICE clients to relay a new chat text message	⟨UPDATE_CHAT_TEXT, TEXT_INFO⟩
CHECK_CONNECTION	Sent from RICE server to all connected RICE clients to check connection validity	⟨CHECK_CONNECTION⟩
END_REQ	Sent from RICE client or RICE server to terminate a connection	⟨END_REQ⟩
END_ACK	Sent from RICE client or RICE server to acknowledge termination of a connection	⟨END_ACK⟩

a control-lock is given to a remote user, we also support the direct control-lock passing to another remote user, without the intervention of the instrument administrator. For this, the RICE client that has the control-lock sends a PASS_LOCK_HOST to the RICE server, which in turn sends a PASS_LOCK to the appropriate RICE client instantly. A similar relaying of chat text between remote users is performed by the RICE server. Here, the chat text from a RICE client is sent to the RICE server using the CHAT_TEXT message, which in turn sends an UPDATE_CHAT_TEXT message to all the other RICE clients. We remark that the relaying of VoIP traffic between the RICE clients is handled by the OpenMCU software using the standard ITU-T H.323 signaling protocols.

In case of any exception conditions, we use a time-out period of 10 seconds. Exception conditions correspond to cases where the RICE server is not running/accessible, or if one or more RICE clients are disconnected due to a network partition during an active session. To detect the latter case, CHECK_CONNECTION messages are sent every 10 seconds from the RICE server to each of the RICE clients. Table 1 summarizes the purpose and format of the various session signaling protocol messages in RICE.

5.3. Applications for research and training

In this section, we describe a few applications of RICE to foster research and training activities involving remote instrumentation. In the context of research, RICE allows multiple experts to jointly collaborate in investigations of samples loaded in scientific instruments. Each of the remote experts can use the quality adjustment slider to easily configure the appropriate frame rates of the image feeds from the instrument to match their respective last-mile network links. During a remote instrumentation session, the self-contained collaboration tools such as VoIP and chat in RICE enable the experts to efficiently communicate with each other. The control-lock passing feature ensures that there are no conflicts in remote operation of the instrument amongst the experts. Using RICE, each expert after completing his/her set of tasks on the instrument can instantly transfer the instrument-control to any other expert who wants to perform another set of tasks on the instrument. Both the control-lock passing and network health monitoring in RICE prevent inadvertent damages to the instrument by the remote users.

In the context of training, all of the above benefits of RICE mentioned for research are equally applicable, except the roles of remote users which are different. A number of students can use RICE to join and participate in a remote instrumentation session that is controlled by an instructor, who also could be remote using RICE. After the training session, the instructor can assign time-slots on the instrument for the students to complete their lab assignments. During the beginning and end of the lab time-slots of the students, the local instrument administrator can pass and retrieve the control-lock amongst the students, respectively.

RICE can be customized for integrating remote instrumentation with scientific web-portals that help in organization and archival of images and datasets acquired from instruments. Specifically, web-services and middleware defined in works such as [9, 29] can be implemented within RICE to automatically upload images and datasets

to the web-portal storage along with basic metadata such as instrument type and date and time stamps. Remote users can then login to the web-portal using their RICE account credentials and supply additional tags and annotations to their respective image and datasets for search and archival functions. If necessary, the users can even submit batch jobs of image processing filters or data analytics to compute clusters via the same web-portal interface. Thus, RICE can be an essential component in cyberinfrastructure deployments that aim at integrating instruments with networking, computing and storage resources for catering to the research and training needs of scientific user-communities.

6. CONCLUSION

In this paper, we modeled and characterized the complex interplay between the user control behavior and video image transfer performance in remote instrumentation sessions. Our remote instrumentation session model borrowed demand and supply terminology from traditional economics and identified the various system states (Idle, Stable, Unstable, Breakdown, and Recovery) and their transition conditions. The transition conditions were found to be primarily driven by time-varying user behavior and connection quality of the network path between the user and the instrument. Analyzing subjective and objective measurements of remote microscopy sessions involving actual users on LAN and WAN network paths, we found that (a) user QoE is highly sensitive to network health fluctuations caused by network congestion, (b) network health impacts both the user's control traffic throughput and microscope's video image transfer rates, and network congestion can hamper user productivity and cause the system to enter into unproductive states (i.e., Unstable and Breakdown states), and (c) the real-time control and video image transfer traffic is extremely bandwidth intensive for achieving "at-the-microscope" QoE. Thus, we demonstrated that remote instrumentation is a demanding network-based immersive multimedia application and is comparable to other applications of its class such as online-gaming, and high-definition videoconferencing.

We also described our RICE software functionalities that leverage our user and network interplay studies to cope with network bottlenecks, and cater to the requirements of remote observation and remote operation use-cases. In particular, we described in detail two main RICE functionalities that facilitated reliable and efficient remote instrumentation. The first functionality corresponded to the network health monitoring coupled with network performance anomaly detection using a "plateau-detector algorithm" that warns and blocks user's control-actions during network congestion periods. The second functionality corresponded to the "session-signaling protocol" we developed to enable multiuser collaboration in RICE using VoIP, chat, presence, and control-lock passing. Finally, we presented potential applications of RICE for research and training purposes that require remote instrumentation capabilities.

ACKNOWLEDGMENTS

This work has been supported in part by The Ohio Board of Regents and The Ohio State University's CAMM VIM program. The authors thank Dr. Peter Collins, Daniel Huber, Robert Williams, and Professor Hamish Fraser of OSU's CAMM for their participation in the remote microscopy testbed, RICE development feedback, and for providing microscope sample graphics used in this paper. They also thank Gordon Renkes (Department of Chemistry, The Ohio State University) for RICE testing, development feedback, and screenshots of RICE in active sessions. Further, they thank Dr. Dave Hudak, Neil Ludban, Dr. Eylem Ekici, Dr. Dong Xuan, and Dr. Steve Gordon for their useful suggestions during the course of the study. A preliminary version of this paper has appeared in the proceedings of ICST/ACM Immersive Telecommunications Conference (IMMERSCOM), 2007 [30].

REFERENCES

[1] "Chemistry Research Instrumentation and Facilities: Departmental Multi-User Instrumentation (CRIF:MU)," Program Solicitation NSF 07-552, National Science Foundation, Arlington, Va, USA, 2007.

[2] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," *IEEE Internet Computing*, vol. 2, no. 1, pp. 33–38, 1998.

[3] K. W. Hodapp, J. B. Jensen, E. M. Irwin, et al., "The Gemini Near-Infrared Imager (NIRI)," *Publications of the Astronomical Society of the Pacific*, vol. 115, no. 814, pp. 1388–1406, 2003.

[4] K. Jeffay, T. Hudson, and M. Parris, "Beyond audio and video: multimedia networking support for distributed, immersive virtual environments," in *Proceedings of the 27th Euromicro Conference (EUROMICRO '01)*, pp. 300–307, Warsaw, Poland, September 2001.

[5] K. Furuya, M. Tanaka, K. Mitsuishi, et al., "Public opened internet electron microscopy in education field," in *Proceedings of the Microscopy and Microanalysis Conference*, vol. 10, pp. 1566–1567, Savannah, Ga, USA, August 2004.

[6] J. Kao, D. Troxel, and S. Kittipiyakul, "Internet remote microscope," in *Telemanipulator and Telepresence Technologies III*, vol. 2901 of *Proceedings of SPIE*, pp. 90–100, Boston, Mass, USA, November 1996.

[7] M. A. O'Keefe, B. Parvin, D. Owen, et al., "Automation for on-line remote-control in-situ electron microscopy," in *Proceedings of the Pfefferkorn Conference on Electron Image and Signal Processing*, Silver Bay, NY, USA, May 1996.

[8] M. Hadida, Y. Kadobayashi, S. Lamont, et al., "Advanced networking for telemicroscopy," in *Proceedings of the of the 10th Annual Internet Society Conference on Advanced Networking for Telemicroscopy (INET '00)*, Yokohama, Japan, July 2000.

[9] T. E. Molina, G. Yang, A. W. Lin, S. T. Peltier, and M. H. Ellisman, "A generalized service-oriented architecture for remote control of scientific imaging instruments," in *Proceedings of the 1st International Conference on e-Science and Grid Computing*, vol. 2005, pp. 550–556, Melbourne, Australia, December 2005.

[10] Ultra VNC Open-source Remote Access Solution, http://www.uvnc.com/.

[11] Real VNC Open-source Remote Access Solution, http://www.realvnc.com/.

[12] Thinklogical KVMoIP Remote Access Solution, http://www.thinklogical.com/.

[13] Avocent KVMoIP Remote Access Solution, http://www.avocent.com/.

[14] Adder KVMoIP Remote Access Solution, http://www.adder.com/.

[15] M. C. Wright, C. R. Hubbard, R. Lenarduzzi, and J. Rome, "Internet-based remote collaboration at the neutron residual stress facility at HFIR," in *Proceedings of the Workshop on New Opportunities for Better User Group Software (NOBUGS '02)*, Gaithersburg, Md, USA, November 2002.

[16] R. H. Cockrum, D. L. Clark, and S. T. Kelly, "Remote internet instrumentation for monitoring ocean data," in *Proceedings of the MTS/IEEE Conference and Exhibition (OCEANS '01)*, vol. 2, pp. 1176–1182, Honolulu, Hawaii, USA, November 2001.

[17] The Ohio State University CAMM VIM Program, http://www.camm.ohio-state.edu/.

[18] S. Winkler, *Digital Video Quality: Vision Models and Metrics*, John Wiley & Sons, New York, NY, USA, 2005.

[19] P. Calyam, D. Krymskiy, M. Sridharan, and P. Schopis, "TBI: end-to-end network performance measurement testbed for empirical bottleneck detection," in *Proceddings of the 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom '05)*, pp. 290–298, Trento, Italy, February 2005.

[20] S. Shalunov and B. Teitelbaum, "One-way Active Measurement Protocol (OWAMP) Requirements," *IETF RFC 3763*, 2004.

[21] J. Mullin, L. Smallwood, A. Watson, and G. Wilson, "New techniques for assessing audio and video quality in real-time interactive communications," ETNA Project Report, IHM-HCI, Lille, France, 2001.

[22] ITU-T Recommendation G.114, "One-Way Transmission Time," 1996.

[23] P. Calyam, M. Sridharan, W. Mandrawa, and P. Schopis, "Performance measurement and analysis of H. 323 traffic," in *Proceedings of the 5th International Workshop on Passive and Active Network Measurement (PAM '04)*, pp. 137–146, Antibes Juan-les-Pins, France, April 2004.

[24] Open H.323 Project, http://sourceforge.net/projects/openh323.

[25] A. Tirumala, L. Cottrell, and T. Dunigan, "Measuring end-to-end bandwidth with Iperf using Web100," in *Proceedings of the 4th International Workshop on Passive and Active Network Measurement (PAM '03)*, La Jolla, Calif, USA, April 2003.

[26] A. B. Downey, "Using pathchar to estimate Internet link characteristics," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '99)*, vol. 27, no. 1, pp. 222–223, Atlata, Ga, USA, May 1999.

[27] A. McGregor and H.-W. Braoun, "Automated event detection for active measurement systems," in *Proceedings of the Workshop on Passive and Active Measurements (PAM '01)*, Amsterdam, The Netherlands, April 2001.

[28] C. Logg, L. Cottrell, and J. Navratil, "Experiences in traceroute and available bandwidth change analysis," in *Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality*, pp. 247–252, Portland, Ore, USA, August-September 2004.

[29] I. Atkinson, D. du Boulay, C. Chee, et al., "Developing CIMA-based cyberinfrastructure for remote access to scientific instruments and collaborative e-research," in *Proceedings of the 5th Australasian Symposium on ACSW Frontiers*, vol. 249 of *ACM International Conference Series*, pp. 3–10, Ballarat, Australia, January-February 2007.

[30] P. Calyam, N. Howes, A. Kalash, and M. Haffner, "User and network interplay in internet telemicroscopy," in *Proceedings of the 1st ICST/ACM International Conference on Immersive Telecommunications (IMMERSCOM '07)*, Verona, Italy, October 2007.