Optimizing the use of X and VNC protocols for support of remote observing

Robert Kibrick[a], Steven L. Allen[a], Al Conrad[b], Terry Stickel[b], and Gregory D. Wirth[b]

[a]UCO/Lick Observatory, University of California, Santa Cruz, CA 95064 USA
[b]W. M. Keck Observatory, Kamuela, HI 96743 USA

ABSTRACT

Remote observing is the dominant mode of operation for both Keck Telescopes and their associated instruments. Over 90% of all Keck observations are carried out remotely from the Keck Headquarters in Waimea, Hawaii (located 40 kilometers from the telescopes on the summit of Mauna Kea). In addition, an increasing number of observations are now conducted by geographically-dispersed observing teams, with some team members working from Waimea while others collaborate from Keck remote observing facilities located in California. Such facilities are now operational on the Santa Cruz and San Diego campuses of the University of California, and at the California Institute of Technology in Pasadena.

This report describes our use of the X[1] and VNC[2] protocols for providing remote and shared graphical displays to distributed teams of observers and observing assistants located at multiple sites. We describe the results of tests involving both protocols, and explore the limitations and performance of each under different regimes of network bandwidth and latency. We also examine other constraints imposed by differences in the processing performance and bit depth of the various frame buffers used to generate these graphical displays.

Other topics covered include the use of *ssh*[3] tunnels for securely encapsulating both X and VNC protocol streams and the results of tests of ssh compression to improve performance under conditions of limited network bandwidth. We also examine trade-offs between different topologies for locating VNC servers and clients when sharing displays between multiple sites.

Keywords: remote observing, X protocol, VNC protocol, ssh protocol

1. INTRODUCTION

Over the past several years, the Internet-2 initiative has spurred significant improvements in network infrastructure throughout the U.S. As a result, many Keck telescope observing programs can now be conducted remotely from any one of several Keck remote observing facilities located in California. Details of the motivation and planning for those facilities and the software architecture on which they are based were discussed in several previous reports.[4–7] Key aspects of those reports are excerpted in Sections 1.4 and 1.5 below to provide the context for the sections that follow them.

1.1. The Keck Remote Observing Model

Keck's remote observing model was first adopted in 1996 and is based on two fundamental principles: 1) all observing applications (including user interfaces) run on summit control computers, and 2) the displays generated by those applications are re-directed to the remote sites from which the observers operate (See Fig. 1). This model provides a generic method for supporting remote operation of a large variety of both optical and infrared instruments, and ensures that all observers, regardless of the location from which they observe, run identical versions of the observing applications.

Further author information- (Send correspondence to R.K.)
R.K.: Email: kibrick@ucolick.org; http://www.ucolick.org/~kibrick; Telephone: 831-459-2262; Fax: 831-459-2298;
S.A.: Email: sla@ucolick.org; Telephone: 831-459-3046; Fax 831-459-2298;
A.C.: Email: aconrad@keck.hawaii.edu; Telephone: 808-881-3812; Fax: 808-885-4464;
T.S.: Email: tstickel@keck.hawaii.edu; Telephone: 808-885-7887;
G.W: Email: wirth@keck.hawaii.edu; Telephone: 808-885-7887

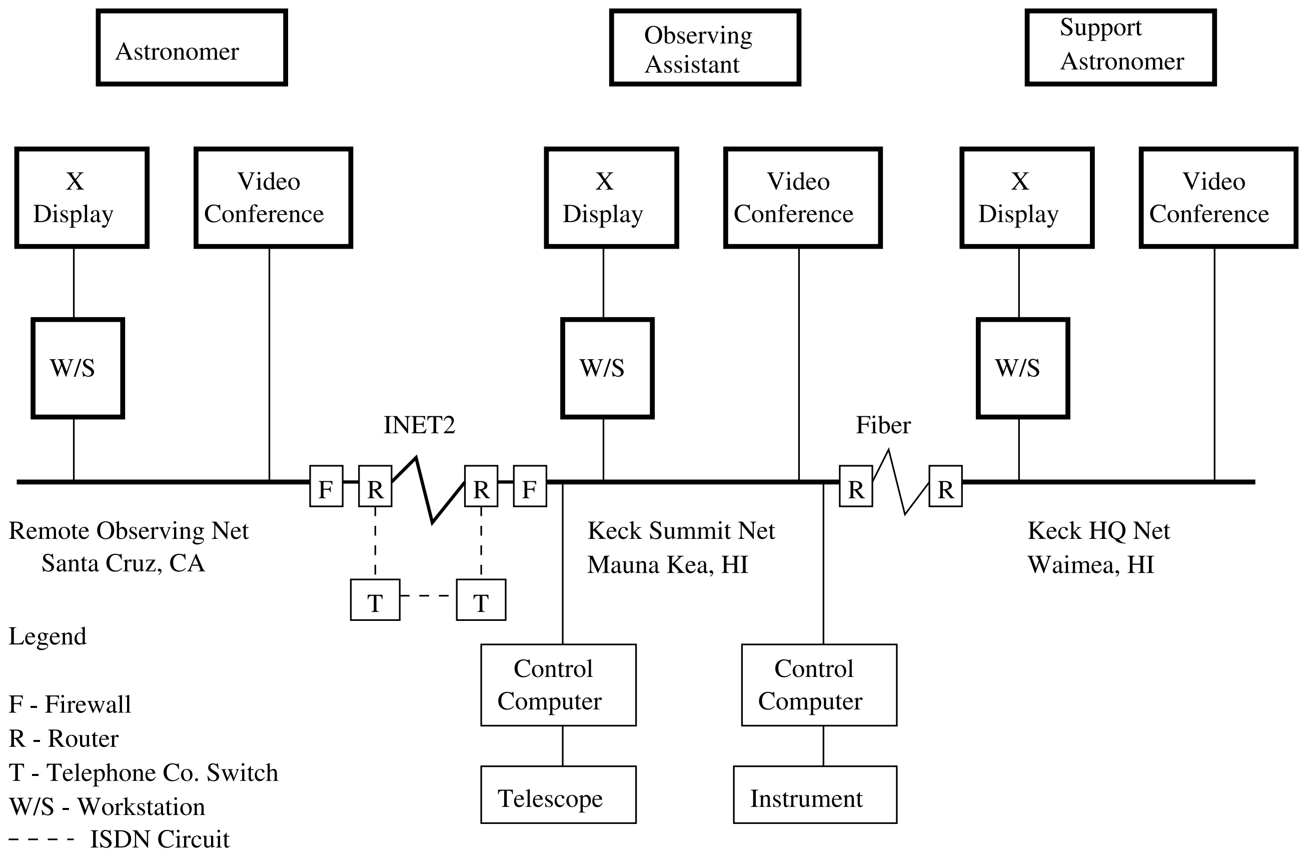

Figure 1. Global overview of connectivity between Santa Cruz, Mauna Kea, and Waimea

1.2. Motivation

The primary motivation for this approach is to minimize the overall level of technical and administrative support required to maintain the operation of observing software at Keck. Under this model, only the single copy that runs on the summit control computers needs to be maintained. As a result, it also reduces the support burden at the remote sites, since the workstations used for remote observing only require installation and maintenance of generic software (e.g., the X Window system and VNC). This is of particular concern, given current efforts to reduce, consolidate, and centralize information technology support on each U.C. campus; those efforts are driven by budget cutbacks that are a result of California's multi-billion-dollar State budget deficits.

In addition, since different Keck instruments were developed by different institutions, there are significant differences in the way in which they operate and in the underlying communication protocols (e.g., RPC,[8] EPICS channel access,[9] MUSIC[10]) that they use. Some of those protocols are sensitive to packet throughput, latency, and jitter issues that are difficult to control across wide area networks (WAN), while others pose challenges in cases where firewalls need to be traversed. By requiring all observing software to run at the summit, these protocol issues are confined to the summit network. This simplifies the task of remote operation in that these WAN-related issues only need to be addressed for generic protocols, such as *ssh*, X and VNC.

1.3. Goals of the current effort

While the Keck remote observing model provides the advantages noted above, it often does not deliver optimal interactive performance to all remote sites. In most cases, it delivers performance that is quite acceptable, but in a few cases interactive response is sluggish at best. A major goal of our current and ongoing effort is to experiment with various combinations of generic protocols to determine which ones provide the best overall performance over the widest range of circumstances. An additional goal of this effort is to enable summit observing support staff to monitor the operations conducted by observers from the remote sites.

1.4. Remote Observing from Keck Headquarters in Waimea

Most observations with the Keck telescopes are now carried out from one of the two remote operations rooms located at Keck Headquarters in Waimea (see Fig. 2). The workstations in these rooms communicate with instrument and telescope control computers at the summit via a dedicated, 45 Mbit/sec fiber link (See Fig. 1).

The observer is supported by an observing assistant (OA), who operates the telescope, and a support astronomer (SA), who assists the observer in setting up and operating the instrument. During the first part of the night, a SA is present in the remote operations room and during the latter part an on-call SA can be reached by telephone at home. The OA is usually located at the summit, but in some cases will operate the telescope from the same control room in Waimea from which the observer is running the instrument. A video-conferencing system links each remote operations room in Waimea with its corresponding telescope control room at the summit.

Figure 2. Keck remote operations room in Waimea

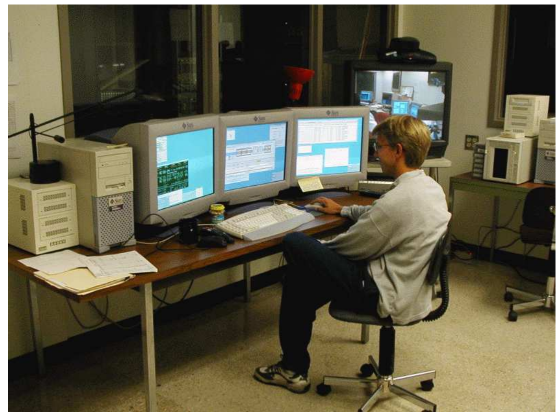

Figure 3. Santa Cruz remote observing room

1.5. The Keck Remote Observing Facilities in California

The remote observing facilities in California (see Fig. 3) are primarily targeted towards observers scheduled for short-duration observing runs and who live within commuting distance of one of those facilities. They are not intended to duplicate the Waimea facility nor to operate independently of it. Rather, each is an extension of the facility in Waimea. That facility and those on the mainland are intended to operate in collaboration, sharing resources where practical. We rely on the existing instrument support staff at Waimea and provide video-conferencing and shared software environments so that they can most effectively support the observers at the mainland sites.

In some cases, an observing run may be conducted by an observing team in which some of its members observe from Waimea while others elect to observe from the mainland. If an observing team is composed of observers from widely separated mainland sites (e.g., Santa Cruz and San Diego), then simultaneous remote operation for multiple mainland sites may prove desirable.

A key objective of the mainland facilities is to provide observers with the flexibility to choose the observing site that is most practical and productive for them. That choice depends on the needs of the members of their observing team and the specific circumstances and duration of a given observing run. That objective can best be achieved if the mainland sites are operated in close collaboration and cooperation with the Waimea facility.

2. USING X PROTOCOL TO REDIRECT DISPLAYS TO REMOTE LOCATIONS

In 1996, Keck initiated remote observing from its headquarters in Waimea. That initial implementation[4] relied exclusively on the use of the X protocol to redirect to Waimea the displays generated by observing client applications running on the summit computers (See Fig. 4). Given the relatively high bandwidth and low latency (under 1 millisecond) between the two sites, this scheme worked relatively well and provided comparable interactive performance at both sites. Its main limitation was that multiple instances of a given client application needed to be run if the display from that application was needed at both sites. In the case of single-copy applications, their displays could be directed to one site or the other, but not simultaneously to both.

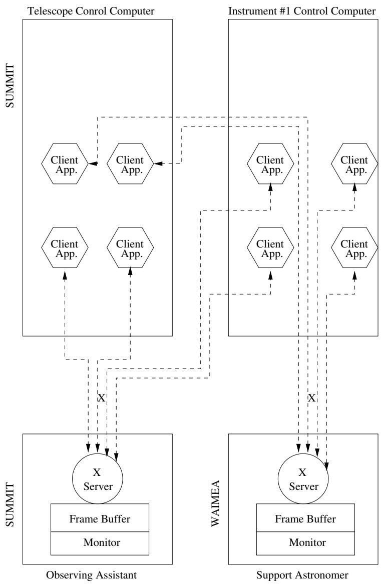

Figure 4. Redirect displays to Waimea with X

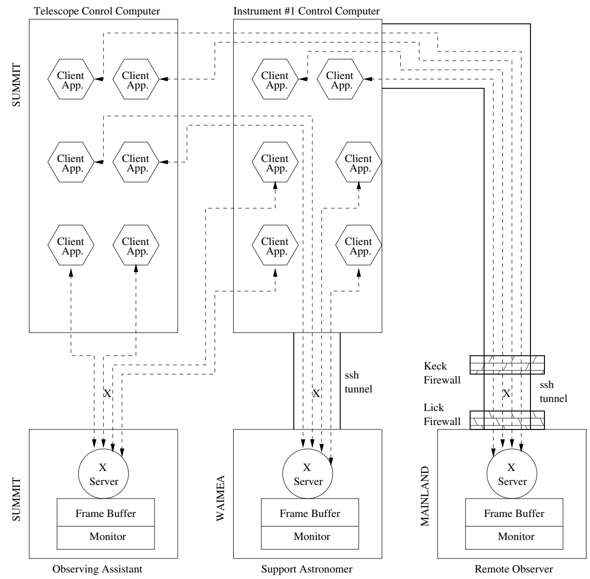

Figure 5. Redirect displays using X protocol

2.1. Security and tunneling

In 1997, experiments began to extend this model to include sites on the mainland[5, 6] (See Fig. 5). That extension posed two significant challenges. First, the X protocol is not secure over WANs, and for that reason, it is blocked by most firewalls. Our solution to that problem was to encapsulate the X protocol traffic within a secure-shell (*ssh*[3]) tunnel. This offers the added advantage of simplifying the handling of the X `DISPLAY` environment variable and eliminating the need for awkward (and less secure) mechanisms such as *xauth* and *xrsh*. Instead, *ssh* provides automatic mechanisms for transparently and securely redirecting the X display from the machine running the client applications to the X server at the remote site.

2.2. Latency and bandwidth

The second challenge encountered in extending this model to the mainland is that packet round-trip times between California and the Mauna Kea summit are over a factor of 100 longer than those between Waimea and the summit. This significantly higher latency profoundly affects remote interactive performance in two ways: 1) it reduces the effective bandwidth of the connection, and 2) it greatly increases the time required for X-based operations that require multiple, sequential round-trip transactions between the X client and server.

The reduction in effective bandwidth that occurs over high latency links is a function of the default TCP window size*. That default typically assumes a low-latency connection (e.g., a LAN connection) and provides for efficient transport of data in such an environment. The effective bandwidth that can be delivered over a TCP/IP channel is set by the window size divided by the round trip time (RTT). For a default window size of 8 Kbytes (the default under Solaris 2.7) and an RTT of 100 milliseconds (the average RTT between Mauna Kea and the mainland), the effective bandwidth will be limited to 80 Kbytes/second, or 0.64 Mbits/second; that is well below the 30 Mbits/second capacity of the slowest leg of the link. While higher bandwidth could be achieved by careful tuning of the TCP parameters on the computers at each of the sites involved in remote operations, that requires detailed coordination between systems administrators at each of these independently managed sites – something that has so far proven difficult to achieve in practice.

*For details, please see: http://www.psc.edu/networking/perf_tune.html and http://www.web100.org/

2.3. In-stream compression

Another approach to compensating for the reduced effective bandwidth is to employ compression on the X protocol streams sent to the remote sites. The most straightforward option for accomplishing this is to enable *ssh* to provide in-stream compression (*ssh -C -X*). The compression that *ssh* provides is not specifically optimized for X protocol streams. Other compression schemes that are include *LBX*[†] and *dxpc*[‡]. While both of those schemes were developed for transmitting X protocol streams across relatively low-bandwidth links, such as analog telephone and ISDN lines, one might assume they could provide some benefit when transmitting such streams across moderate-bandwidth high-latency links such as the one between Mauna Kea and the mainland.

Tests with *dxpc* yielded compression factors of 5 to 7 during startup of X-based GUIs. However, this dramatic reduction in the volume of X protocol data transmitted yielded only a 10% reduction in GUI startup time for client GUIs on Mauna Kea displaying to X servers in California. Similar results were obtained using *ssh* with compression enabled. Packard and Gettys,[11] in their extensive analysis of X Window system network performance, evaluated the utility of compressing X protocol streams sent over high latency links and compared the relative effectiveness of the compression provided by *LBX* and *ssh*. They concluded that when using X protocol over links with 100 millisecond RTTs, the performance with *LBX* was no better than when using an *ssh* tunnel with compression enabled. They also concluded that *LBX* solved neither the authentication nor security problems inherent to networked distribution of X protocol streams, and concluded that *LBX* was an inferior alternative to *ssh*. Based on the measurements they conducted, they also concluded that for high latency links (such as the one between Mauna Kea and California), the effects of latency usually dominate those of bandwidth during application start up.

2.4. Latency-sensitive operations prolong application startup

The long RTT (100 milliseconds) between Mauna Kea and the mainland negatively impacts any X-based graphical operations (e.g., initial creation and display of a window, visualization of a pulldown menu, etc.) that require multiple, sequential round-trip transactions between the client application and the X server responsible for performing that operation. As a result, operations that complete in a fraction of second when initiated from an X server running on a workstation in Waimea instead require several seconds when initiated from an X server running in California. Because this sluggish performance is primarily due to the propagation delays between the sites, increasing the effective bandwidth (by upgrading the inherent bandwidth of the link itself, by tuning of TCP/IP parameters, or by compressing the X protocol stream) does not address this problem.

One response to this problem is to develop GUIs that minimize the use of those operations that require repeated round-trip transactions between the summit client and the California X server (e.g., creation of pop-up windows), or to perform these expensive operations during initial session setup and before observing commences. For example, one of our GUIs makes extensive use of pop-up windows. We modified it to support a "remote mode" in which it creates and then iconifies all its pop-ups during application startup. While this may take as long as two minutes, it saves times later during observing. Provided that backing store is enabled in the X server, a subsequent instantiation of any of these iconified pop-up windows merely de-iconifies that window, rather than incurring the expense of recreating it. When the pop-up is dismissed, it is simply re-iconified rather than deleted. Such iconfication and de-iconification occurs almost instantly and thus significantly improves observing efficiency.

Unfortunately, this approach is not applicable to legacy GUIs that are no longer actively maintained and which cannot cost-effectively be modified to operate in this manner. Nor is it applicable to other legacy applications, such as the Keck autoguider eavesdrop display, which unnecessarily makes repeated transactions with the X server on every frame update. The unacceptably sluggish performance of such legacy applications when their displays are redirected to California was one of the primary motivations for exploring VNC as an alternative method for providing remote displays. VNC also enables us to share single-copy applications that cannot be effectively shared using the X protocol by itself.

[†] Low Bandwidth X. For details, see http://www.tldp.org/HOWTO/LBX.html

[‡] Differential X Protocol Compressor. For details, see http://www.vigor.nu/dxpc

3. USING VNC PROTOCOL TO REDIRECT DISPLAYS TO REMOTE LOCATIONS

Virtual Network Computing (VNC[2]) is a thin-client system in which a generic VNC client (the VNC viewer, with which the user interacts) offers remote access to a shareable virtual desktop environment provided by a VNC server; multiple VNC viewers can be connected to a single, shared VNC server. Because VNC viewers store no recoverable state information, they can connect and disconnect from the VNC server with no effect on the session at the server. The VNC protocol works at the framebuffer level, and is based on a single graphics primitive: put a rectangle of pixel data at a given x,y position. Because it works at this low level, VNC servers and clients can be implemented for most operating systems and windowing systems.

VNC offers the potential for solving two of the most vexing problems posed by an exclusively X-based model for remote observing: 1) the sluggish response provided by those X-based GUIs that repeatedly perform operations sensitive to high network latency, and 2) the inability to share between sites those applications for which only a single copy can be invoked.

3.1. Reducing the effects of network latency

The first solution is the less obvious of the two. As noted earlier, the sluggish performance observed with some X-based GUIs (especially when first starting up) results from from the high latencies encountered with each and every round-trip transaction that occurs between a widely-separated X client and server. Those latencies would become negligible if both the X client and server were running on the same machine, or at least on the same network segment. VNC makes that possible, because the VNC server (which acts as a shared, virtual X server) is run on a computer at Mauna Kea (see Fig. 6).

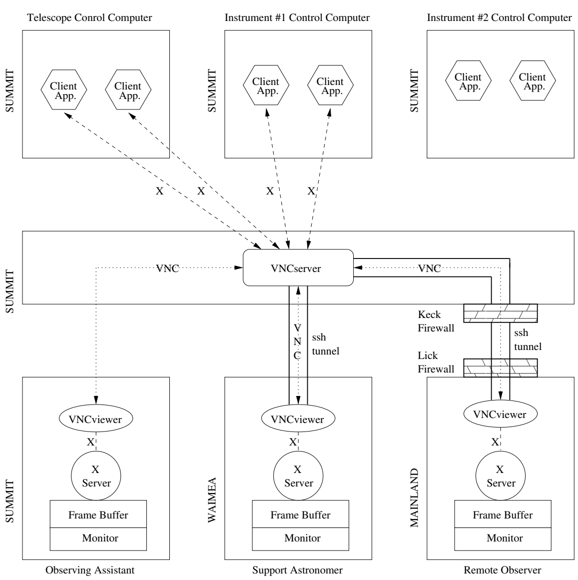

Figure 6. Re-direct displays using VNC protocol

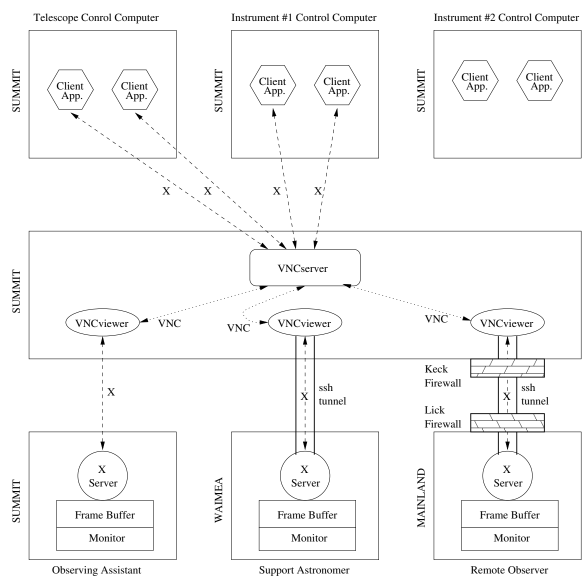

Figure 7. Run VNC server and viewers at summit

Thus, when an X-based GUI client on a summit control computer establishes a connection to its X server (in this case, the VNC server), that server is on the local network and the latencies are minimal. Rather than sending X protocol packets across the Pacific Ocean, we now send VNC protocol packets instead. If the VNC protocol requires fewer round-trip transactions to transmit the corresponding display updates between the VNC server on Mauna Kea and the VNC viewer in California, then the time required for GUI startup should be significantly reduced and the responsiveness of latency-sensitive GUI operations (e.g., popping up new windows) should be greatly increased. This is what we find in practice.

For example: suppose one uses the arrangement depicted in Fig. 5 and starts a *ds9*[§] image display program on the summit instrument control computer with the X display redirected to an X server in California via an

[§]See http://hea-www.harvard.edu/RD/ds9/

ssh tunnel with in-stream compression enabled. The application then requires between 70 to 75 seconds to startup and complete painting the remote display. If the arrangement depicted in Fig. 6 is used instead (i.e., ds9's X display is redirected to the local VNC server which relays the display updates to a VNC viewer via a port-forwarded and compression-enabled *ssh* tunnel), the application starts up and completes painting the remote display in approximately 11 seconds.

Unfortunately, while the VNC topology depicted in Fig. 6 significantly reduces the time required for application startup, other aspects of interactive performance are significantly degraded compared to using the X protocol topology depicted in Fig. 5. For example, when using X, iconify/de-iconify operations are nearly instantaneous (because of local backing store), while when using VNC, these same operations take tens of seconds. Similarly, when using X, *ds9* zoom in/out operations on a 8K x 8K pixel image complete in 4 seconds, while using VNC, they require 9 seconds to complete. (Such operations primarily push data in one direction and are thus more sensitive to effective bandwidth than to network latency.) In addition, X provides superior interactive performance in terms of scrolling the *ds9* color map and using the mouse to generate projections and regions.

3.2. Security and tunneling

Like X, VNC is not sufficiently secure over WANs and requires the use of a tunnel to traverse firewalls. Unlike the support it provides for X packets, *ssh* provides no built-in capability for forwarding VNC protocol packets. In order to establish a secure tunnel between the VNC server on Mauna Kea and the VNC viewer in California, the port-forwarding capability of *ssh* (*ssh -L*) must be used. The port-forwarding tunnel must be established before the VNC viewer in California attempts to connect to the VNC server on Mauna Kea, and this tunnel must be kept open for the duration of the VNC connection. Should the tunnel collapse, there may be a several-minute-long TCP-imposed time delay before the tunnel can be re-established for that port; that would be troublesome if it occurred during observing.

3.3. Shared environment

The most significant advantage of VNC is its ability to enable the sharing of applications between multiple sites. This is especially important for those applications which cannot otherwise be shared. In addition, since all of the VNC viewers connected to the same VNC server see the identical shared desktop, users at a given site can directly observe the mouse motions and keystrokes of their colleagues at the other sites. This has proven to be an extremely valuable capability both for training and troubleshooting.

For example, a support astronomer (SA) in Waimea might be trying to provide (via the videoconferencing connection) advice to an astronomer on the mainland about a specific function of the instrument GUI. If that GUI is displayed on the shared VNC virtual desktop visible at both sites, the SA can use the shared mouse to point at various features on that GUI while describing their functionality. After providing such a brief tutorial, the SA can then ask the astronomer to perform the desired operation and can then watch the astronomer's keystrokes and mouse motions/actions, to confirm that the astronomer is performing the sequence correctly.

However, the shared desktop can also be a source of confusion. If more than one user types into the shared desktop at the same time, their respective keystrokes will be interleaved, and if both try moving their mice at the same time, they will compete for control of the window focus. If one has not used a shared virtual desktop before, some time is needed to become accustomed to these potential conflicts.

The VNC viewer does provide a read-only mode in which it displays the shared virtual desktop but cannot modify it. This capability is extremely useful in the case where the remote participants simply want to watch what is going on without risk of accidentally affecting the observations in progress.

A final advantage of VNC is that the virtual shared desktop persists on the VNC server regardless of VNC any viewers/clients. If, for some reason, the remote participant's computer crashes, the session on the VNC server will remain intact and no context will be lost. Once the computer at the remote site is rebooted and the VNC viewer restarted and reconnected to the VNC server, the remote observing should continue without any intervention needed on the summit machine.

4. ALTERNATIVE TOPOLOGIES

Alternative topologies were explored to determine whether any offered improved performance over those depicted in Figs. 5 and 6. One such topology, show in Fig. 7, shows considerable promise. Because it retains the use of a VNC server running on a computer on the Mauna Kea summit, the X-based GUI clients that run on the control computer connect to a local X server (i.e., the VNC server) and thus encounter minimal network latency; this significantly reduces application start up time as well as the time required to create new pop-up windows. At the same time, by running the VNC viewers on the same computer as the server, the VNC protocol packets sent between server and viewers can flow over an extremely high-bandwidth Unix domain socket connection and not burden the local network.[¶]. Alternatively, the VNC viewers can be run on another computer on the same local network as the VNC server, and still obtain a very high-bandwidth connection.

The topology depicted in Fig. 7 has the added advantage that the VNC package need only be installed and maintained at the summit and not at each remote site, and thus avoids one more area where software versions could drift out of sync between sites. In addition, it avoids the complexity associated with the port-forwarding features of *ssh*; while most observers are familiar with using *slogin/ssh* to log into a remote computer, most are not familiar with how to forward ports from one machine to another. It also avoids the potential TCP socket-timeout delay problem described in Sect. 3.2.

The interactive performance provided by this topology is roughly comparable to that obtained when running the VNC viewers at the remote sites (i.e., the topology of Fig. 6). However, it provides marginally better performance on *ds9* zoom in/out operations (6.4 seconds versus 9.3 seconds). But for image display applications such as *ds9*, the best interactive performance is still achieved by bypassing VNC altogether and using X protocol to redirect the *ds9* display to California (i.e., Fig. 5); in that mode, *ds9* zoom in/out operations take only 4 seconds. The main penalty one incurs from using X protocol to redirect the display for *ds9* is the excruciatingly long time (72 seconds) required for application startup. Provided that the application only needs to be started once at the beginning of the observing session, this is a small price to pay for the significantly improved interactive performance achieved throughout the remainder of the session.

5. USING A MIX OF BOTH PROTOCOLS

Unfortunately, neither the X protocol nor the VNC protocol provides an optimal solution for all applications or across all network paths. Accordingly, the optimal solution for a given mix of applications whose displays are to be distributed to a particular set of remote sites will likely require a mix of both protocols. Although we are still gathering and analyzing performance measurements, some preliminary results can be summarized here.

First, for image display applications such as *ds9*, the topology depicted in Fig. 5 currently appears to provide the best interactive performance, but the worst application start up time. That topology also provides the best interactive performance for X-based GUIs which iconify (rather than delete) any dismissed pop-up windows, provided that: 1) all of those windows were first popped-up and then iconified at the start of the session and 2) backing store is enabled in the X server.

Second, for those GUIs (e.g., *xlris* and *xpose*) which delete dismissed pop-up windows and recreate them anew each time they are popped-up, the topologies depicted in Figs. 6 and 7 will usually provide better interactive performance than the topology depicted in Fig. 5. These two topologies also provide the best performance for applications (such as the Keck autoguider eavesdrop) which are extremely sensitive to network latency. They also appear to provide nearly optimal performance for output-only applications such as *facsum* and *xmet*[‖]. Of

[¶]To enable VNC server to accept connections on a Unix domain socket, the default protection on the /tmp/.X11-unix/ directory (which is reset whenever the system is rebooted) must be set to enable the server to create a named socket there; root access is required to effect that change. Otherwise, the VNC server must be started with the `"-nolisten local"` option and will then only accept connections on network sockets.

[‖]These applications are specific to Keck: *xlris* is the GUI for the Low-Resolution Imaging Spectrograph (LRIS); *xpose* is the exposure control GUI used by all of the Keck optical CCD systems; *facsum* is the Keck Facilities Summary, which displays the current telescope and dome positions and which can be seen displayed on the leftmost screen in Fig. 2; *xmet* is the status display for the Keck meteorological station.

these two topologies, the one depicted in Fig. 7 is somewhat easier to administer and requires less software to maintain at the remote sites.

The ability to use a mix of X and VNC protocols (for redirecting the displays of a variety of applications) is simplified by the availability of multiple display monitors at each site. As can be seen in Figs. 2 and 3, most remote observing sites have at least 3 monitors available for the display of the various GUIs and image display applications (e.g., *ds9*). For some instruments, two monitors may be configured to provide local desktops onto which remote applications will be displayed directly via X-protocol (i.e., the topology of Fig. 5), while the third monitor will be used to display the shared virtual desktop provided by the VNC server at the summit. For other instruments, two monitors might be configured to display shared virtual desktops, and in other cases all three monitors might be configured to operate in that mode.

It may also prove to be the case that for a given instrument, different topologies may prove optimal for different sites. For example, while the topology of Fig. 7 appears to provide marginally better performance than the topology of Fig. 6 for displays redirected to the mainland, the converse may prove true for displays redirected to Waimea. If so, that difference is probably due to the much smaller network latencies between the summit and Waimea compared to those between the summit and the mainland.[**]

In addition to these various permutations and combinations, there are a number of other specific issues relating to the specific frame buffer hardware and versions of X window software deployed at each of the remote sites. These are explored in the next section.

6. ISSUES AND INTERACTIONS

6.1. X visuals

The underlying hardware of an X server constrains the colormap characteristics which it can offer. In the X Window System these characteristics are known as visuals. Typical visuals are 8, 16, or 24 bits deep and categorized as *PseudoColor*, *StaticColor*, and *TrueColor*. An X client program requires different strategies, and therefore different code, to handle each different visual. A robust X client program should query the X server to determine what visuals are available and (when it cannot use the default visual) choose a visual which it is best able to use.

X programmers often have not had access to all types of X11 visuals. Consequently many X clients are unable to run on some X servers. Fortunately it is now becoming more common for the X servers supplied by hardware vendors to support multiple simultaneous visuals. Often it is possible to configure such X servers to offer a particular visual as the default. Having multiple visuals available simultaneously permits advanced X clients to choose the visual most suitable for themselves; it can also permit other X clients which do not know how to choose between multiple visuals to obtain the most suitable visual by default.

The Xvnc server, unfortunately, is not currently able to offer multiple simultaneous visuals. Xvnc must be started specifying a visual which will be acceptable to all of the X clients which will be on the shared virtual desktop. This means that it is not possible for Xvnc to provide the best service to all possible clients. If some X clients require 8-bit *PseudoColor* and others prefer 24-bit *TrueColor*, then it might be necessary to run two different VNCservers with the two different visuals, then run pairs of vncviewer programs to display onto two different remote sessions at each remote site.

For remote observing we use some legacy image display programs and some GUIs (developed with *DataViews*) which can only operate on 8-bit *PseudoColor* visuals. Therefore we are typically constrained to run Xvnc with the switch *-cc 3* (to specify *PseudoColor*) and the switch *-depth 8*.

[**]For any mainland site, the strategies discussed here cannot eliminate a perceptible delay in response to user keyboard/mouse events; even with optimal topologies, some patience will be required of remote observers.

6.2. Colormap issues

When using VNC, the *vncviewer* program is constantly translating the color requests suitable for the Xvnc virtual server into color requests suitable for the real X server where the user sees the shared session. For example, in the case of an X server with an 8-bit *PseudoColor* visual, an image display program can simply update the colormap without rewriting any pixels in the frame buffer. Updating the colormap typically gives an instantaneous change in the appearance of a displayed image on a real X server. But when this is an Xvnc virtual server the colormap change often requires the *vncviewer* program to redetermine and retransmit the value of every pixel in the display of the shared session. The recoloring consumes CPU time on the machine running *vncviewer* and the retransmission consumes bandwidth between the virtual and real X servers. As a result, whether or not the visual characteristics in the virtual and real X server match, an update in the appearance of the image can be much slower.

The *vncviewer* program has numerous options which modify its use of visuals and colormaps on the real X server. There are advantages and disadvantages to these colormap options. Some of our remote observing sites run X servers offering 24-bit *TrueColor* visuals (often in combination with other simultaneously available visuals). When *vncviewer* chooses a 24-bit *TrueColor* visual it guarantees that the colors on the real X server will match what is expected, but the cost is the bandwidth required to transmit 24 bits for every pixel that changes value.

Users of X servers with 8-bit *PseudoColor* are familiar with the "colormap flashing" that occurs when multiple X client programs request more colors than are available. In such cases the color-greedy X clients often specify a private color map which is used when the window manager gives focus to their windows. The *vncviewer* offers the option -owncmap which attempts to choose an 8-bit PseudoColor visual for the window displaying the shared session on the real X server. In the ideal case this option permits the VNC server and client programs to update images by changing only the colormap. However, the private colormap for the virtual session may make it impossible to view any other X clients on the server outside of the window with the shared session.

In the absence of the *-owncmap* option the vncviewer may not be able to allocate many colormap entries on a PseudoColor X server. In that case vncviewer will choose colors which it deems to be close to the desired color, but the results are often so different that it becomes difficult to recognize and use some GUIs.

6.3. *whitepixel* and *blackpixel* issues

Every X server reserves two contrasting values in its colormap with the designations blackpixel and whitepixel. Typically the pixel values for whitepixel and blackpixel are chosen to be 0 and 1, but there is no guarantee which value will be which color. Although X clients should query the server to determine these two pixel values, not all do so.

When running via Xvnc an X client program is always dealing with at least two different X servers. One is the Xvnc shared virtual server, and the others are each of the X servers onto which vncviewer is duplicating the shared session. In some cases the Xvnc virtual server and the X servers where the shared session users sit choose different pixel values for white and black. In such cases we observe that some X client programs can cause problems such as white text on white background, or black on black. We have attempted several different remedies to this complex situation.

The whitepixel and blackpixel values are often configurable. In a fashion that was once common among many X servers, the Xvnc virtual server permits some control over the blackpixel and whitepixel values using undocumented[††] command line switches *-whitepixel* and *-blackpixel*. Newer X servers tend to permit somewhat less flexibility with a command line switch *-flipPixels* that allows swapping of the black and white pixel values.

In some cases it may be possible to use these switches to remedy problems with the choice of whitepixel and blackpixel on the various servers, but this is both difficult and not guaranteed to solve the problems. Whereas the choice of the Xvnc pixel values can easily be made at the start of a virtual session, the users sitting at remote session points may not have the administrative privileges necessary to reconfigure their X servers. Moreover, a multi-headed X server using several different kinds of graphics cards may provide different values of blackpixel

[††]See http://www.realvnc.com/pipermail/vnc-list/2001-August/024147.html

and whitepixel on each of its screens. Unless all remote X servers can be configured to provide a screen where the pixel values agree with the Xvnc virtual server, this strategy will probably fail.

Rather than struggle with variant X servers and broken X clients, a simpler strategy can usually avoid black on black and white on white. For any X clients which exhibit this behavior, simply configure them by requesting foreground and background colors other than white and black; e.g., white on blue, or yellow on black. Choosing colors other than black and white comes at the price of consuming at least one more colormap entry, but it is much quicker than determining whether there is any configuration of X servers which can avoid the problem.

6.4. Font issues

In the X Window System the fonts available to an X client program are usually supplied by the X server. In the case of VNC, this means that the available fonts are the ones available on the machine running vncserver. These are the same fonts that would be available to a native X server on that machine either through its own file system or via an X font server. In general the use of fonts via VNC will be less efficient than on a native X server because the rendered glyphs must be transmitted over the wire. The use of VNC can be an advantage if the X clients which are to be duplicated require peculiar fonts which are not readily available at the remote sites, but in such a case it will be necessary to guarantee that Xvnc is made aware of those fonts.

7. MEASUREMENTS OF REMOTE PERFORMANCE

Table 1 summarizes the results of timing tests conducted for each of the three topologies described in Sects. 2-4. For each topology, measurements were made both with and without compression of the protocol stream sent over the *ssh* tunnel. The *ds9* image display program was used at the test case client program because it is readily available and provides a representative range of functions to test. For comparison, the times required to perform these functions using a *ds9* run from another host on the Santa Cruz network are listed in the "local" column.

These measurements were made at times when the computers at the tunnel endpoints were otherwise idle (as verified by running the *top* utility) and during times of relatively low utilization of the network link between Mauna Kea and California (as verified by monitoring Internet traffic statistics). The control computer at the Mauna Kea endpoint was a Sun Sparc Enterprise 450 with four 300 MHz CPUs running Solaris 2.8, *ssh* version OpenSSH_3.6.1p2 and Xvnc version 3.3.3r1; it ran both the client application (i.e., *ds9*), and, where applicable, the VNC server and viewer. The computer in Santa Cruz was a Sparcstation 10 with one 440 MHz CPU running Solaris 2.7, *ssh* version OpenSSH_2.9.9p2, and Xvnc version version 3.3.3r1; in the Fig. 6 topology, it also ran the VNC viewer. Both computers used a default TCP window size of 8192 bytes.

Test sequence	local	Fig. 5 yes	Fig. 5 no	Fig. 6 yes	Fig. 6 no	Fig. 7 yes	Fig. 7 no
VNC viewer startup	N/A	N/A	N/A	10.0	33.0	12.7	32.0
ds9 startup	6.6	72.0	76.0	11.0	27.5	10.6	26.2
file chooser popup	3.0	10.3	11.9	3.0	7.7	3.0	6.8
ds9 zoom in	0.7	4.0	10.2	9.0	17.0	6.5	11.5
ds9 zoom out	0.7	4.5	9.5	9.7	17.5	6.3	13.5
ds9 draw cut	0.0	0.1	3.0	4.0	10.0	4.5	13.5
ds9 make plot	1.0	4.5	9.0	7.0	10.0	4.5	12.0
stats box popup	6.9	13.0	20.0	11.7	20.0	13.8	18.5

Table 1. Response time in seconds for each topology with(yes) and without(no) compression

All measurements were made multiple times and on different days, and the results checked for consistency to weed out any spurious measurements that might be skewed by unexpected bursts of network activity. A better methodology for making such measurements would be to use the NIST network simulator package[12] to provide a consistently repeatable simulation of the Mauna Kea to California network link.

These measurements show that in most cases the use of in-stream compression of the protocol streams significantly improved performance for all three topologies. This suggests that further improvement might be achieved via more optimal tuning of TCP parameters at the endpoints of the tunnel.

8. CONCLUSION

In developing methods to provide access to interactive X-based applications from remote sites, consideration must be given to the constraints imposed by network bandwidth and latency. While the past 6 years have seen a twenty-fold increase in the limiting bandwidth of the link between Mauna Kea and the mainland, the latency of that link (now 100 milliseconds) has also increased, along with the hop count (now 22). The negative effects of that high latency can be partially mitigated by an appropriate choice of transport protocols and software topologies. Interactive response can be optimized by combining the best features of X, VNC, and *ssh*.

ACKNOWLEDGMENTS

We wish to acknowledge the combined efforts of the University of Hawaii, the Gemini Telescope consortium, the National Science Foundation, the Department of Defense, the University Corporation for Advanced Internet Development, and the Corporation for Education Network Initiatives in California toward the establishment of a reliable, high-bandwidth network pathway between the Mauna Kea summit and the U.S. mainland.

We also thank Keck Observatory support astronomers Bob Goodrich and Paola Amico for their assistance in remotely operating the LRIS, ESI, and DEIMOS instruments from Santa Cruz, and software engineers Liz Chock and Julia Simmons for their helpful assistance in resolving software issues. We also acknowledge the efforts of the Lick Observatory Network Information and Computing Systems group, including Jeff Mallory, Kyle Lanclos, and John Rickard, for their assistance in configuring the remote observing facility at UCSC. We thank UCSC campus network engineers Jim Warner and Mark Boolootian, and Keck Observatory system administration manager Jon Chock, for their continuing efforts to keep the networks at each end point reliable and secure. We also acknowledge the efforts of David Tytler, John O'Meara, and Pete James in setting up the Keck remote observing facility at U.C. San Diego, and Patrick Shopbell for his efforts in setting up the facility at Caltech.

REFERENCES

1. X.org Foundation, "X.org home page," *http://www.x.org* , 2004.
2. T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual Network Computing," *IEEE Internet Computing* **2**(1), pp. 33–38, January/February 1998.
3. T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen, "SSH Protocol Architecture," *http://www.openssh.org/txt/draft-ietf-secsh-architecture-12.txt* , 2002.
4. A. R. Conrad, R. Kibrick, and J. Gathright, "Remote Observing with the Keck Telescopes," in *Telescope Control Systems II*, H. Lewis, ed., *Proc. SPIE* **3112**, pp. 99–110, 1997.
5. R. I. Kibrick, S. L. Allen, and A. R. Conrad, "Remote observing with the Keck Telescopes from the U.S mainland," in *Advanced Global Communications Technologies for Astronomy*, R. Kibrick and A. Wallander, eds., *Proc. SPIE* **4011**, pp. 104–116, 2000.
6. R. I. Kibrick, B. Hayes, S. L. Allen, and A. Conrad, "Remote observing with the Keck Telescopes from multiple sites in california," in *Advanced Global Communications Technologies for Astronomy II*, R. I. Kibrick, ed., *Proc. SPIE* **4845**, pp. 80–93, 2002.
7. P. L. Shopbell, R. I. Kibrick, S. L. Allen, and B. Hayes, "Remote Observing on the Keck Telescopes," in *Astronomical Data Analysis and Software Systems XII*, H. E. Payne, R. I. Jedrzejewski, and R. N. Hook, eds., *ASP Conference Series* **295**, pp. 170–173, Astronomical Society of the Pacific, 2003.
8. Sun Microsystems, "RFC 1050 - RPC: Remote Procedure Call Protocol specification," *http://www.faqs.org/rfcs/rfc1050.html* , 1988.
9. J. O. Hill, *EPICS R3.12 Channel Access Reference Manual*, Los Alamos National Laboratory, http://lansce.lanl.gov/lansce8/Epics/ca/cadoc/cadoc_1.htm, 1996.
10. R. Stover, *MUSIC: A Multi-User System for Instrument Control*, vol. 54 of *Lick Observatory Technical Reports*, UCO/Lick Observatory, http://www.ucolick.org/~de/KSD/music.ps, 1991.
11. K. Packard and J. Gettys, "X Window System Network Performance," in *USENIX Annual Technical Conference, FREENIX Track*, pp. 207–218, USENIX, (http://freedesktop.org/~keithp/talks/usenix2003/), 2003.
12. National Institute of Standards and Technology, "NIST Network Emulation Tool," *http://www.antd.nist.gov/tools/nistnet/index.html* , 2000.