Appears in *Proceedings of the ACM SIGMETRICS 2002*, Marina del Rey, CA, June 15-19, 2002.

Limits of Wide-Area Thin-Client Computing

Albert Lai
Department of Computer Science
Columbia University
1214 Amsterdam Avenue MC0401
New York, NY 10027-7003
amlai@cs.columbia.edu

Jason Nieh
Department of Computer Science
Columbia University
1214 Amsterdam Avenue MC0401
New York, NY 10027-7003
nieh@cs.columbia.edu

ABSTRACT

While many application service providers have proposed using thin-client computing to deliver computational services over the Internet, little work has been done to evaluate the effectiveness of thin-client computing in a wide-area network. To assess the potential of thin-client computing in the context of future commodity high-bandwidth Internet access, we have used a novel, non-invasive slow-motion benchmarking technique to evaluate the performance of several popular thin-client computing platforms in delivering computational services cross-country over Internet2. Our results show that using thin-client computing in a wide-area network environment can deliver acceptable performance over Internet2, even when client and server are located thousands of miles apart on opposite ends of the country. However, performance varies widely among thin-client platforms and not all platforms are suitable for this environment. While many thin-client systems are touted as being bandwidth efficient, we show that network latency is often the key factor in limiting wide-area thin-client performance. Furthermore, we show that the same techniques used to improve bandwidth efficiency often result in worse overall performance in wide-area networks. We characterize and analyze the different design choices in the various thin-client platforms and explain which of these choices should be selected for supporting wide-area computing services.

1. INTRODUCTION

Rapid improvements in network bandwidth, cost, and ubiquity combined with the high total cost of ownership of PC desktop computers have created a growing market for application service providers (ASPs). Going beyond just web hosting, ASPs operate, maintain, and deliver customer data and applications from professional-managed data centers, sparing their customers the headache of buying and maintaining their own hardware and software. The ASP market is projected to grow an order of magnitude to $25 billion in worldwide revenue by 2004 [16]. To provide the infrastructure to deliver easier-to-maintain computational services anywhere on the Internet, a growing number of ASPs are embracing a thin-client computing model [5, 12, 26].

A thin-client computing system consists of a server and a client that communicate over a network using a remote display protocol. The protocol allows graphical displays to be virtualized and served across a network to a client device, while application logic is executed on the server. Using the remote display protocol, the client transmits user input to the server, and the server returns screen updates of the user interface of the applications from the server to the client. Many of these remote display protocols can effectively web-enable applications without application modification. Examples of popular thin-client platforms include Citrix MetaFrame [6, 20], Microsoft Terminal Services [9, 21], AT&T Virtual Network Computing (VNC) [25] and Tarantella [29, 32]. The remote server typically runs a standard server operating system and is used for executing all application logic. Because all application processing is done on the server, the client only needs to be able to display and manipulate the user interface. The client can either be a specialized hardware device or simply an application that runs on a low-end personal computer.

While many ASPs have proposed using thin-client computing to deliver computational services over the Internet, little work has been done to evaluate the effectiveness of thin-client computing in a wide-area network (WAN). Thin-client computing vendors often tout the bandwidth efficiency of their platforms, but as network technologies improve and high-bandwidth Internet access becomes a commodity, bandwidth efficiency alone may not be a good measure of wide-area thin-client performance. Existing ASPs have primarily focused on supporting simple office-productivity tools. It is unclear if the remote display approach used in thin-client computing can effectively support the growing class of graphics and multimedia-oriented applications. Because the importance of thin-client computing will only continue to increase with the rapidly growing ASP market, it is crucial to determine the effectiveness of thin-client computing in WANs on the kinds of web-based and multimedia applications that users are already using and will increasingly be using in the future.

To assess the limits of using thin clients to provide wide-area ubiquitous computing, we have characterized the design choices of underlying remote display technologies and measured the impact of these choices on the performance of thin-client computing platforms in delivering computational services cross-country over Internet2. For our study, we considered a diversity of design choices as exhibited by six of the most popular thin-client platforms in use today: Citrix MetaFrame, Microsoft Windows 2000 Terminal Services, AT&T VNC, Tarantella, Sun Ray [28, 31], and X [27]. These platforms were chosen for their popularity, performance, and diverse design approaches. We focus on evaluating these thin-client

Platform	Display Protocol	Display Encoding	Screen Updates	Compression	Max Display Depth	Transport Protocol
Citrix MetaFrame	ICA	Low-level graphics	Server-push, lazy	RLE	8-bit color *	TCP/IP
Microsoft Terminal Services	RDP	Low-level graphics	Server-push, lazy	RLE	8-bit color	TCP/IP
Tarantella	AIP	Low-level graphics	Server-push, eager or lazy depending on bandwidth, load	Adaptively enabled, RLE and LZW at low bandwidths	8-bit color	TCP/IP
AT&T VNC	VNC	2D draw primitives	Client-pull, lazy updates between client requests discarded	Hextile (2D RLE)	24-bit color	TCP/IP
Sun Ray	Sun Ray	2D draw primitives	Server-push, eager	None	24-bit color	UDP/IP
X11R6	X	High-level graphics	Server-push, eager	None	24-bit color	TCP/IP

* Citrix MetaFrame XP now offers the option of 24-bit color depth, but this was not available at the time of our experiments.

Table 1: Thin-client computing platforms

platforms with respect to their performance on web and multimedia applications, which are increasingly populating the computing desktop. We conducted our experiments using Internet2 because it provides the kind of high-bandwidth network access that we expect will become increasingly cost-effective and accessible in future WAN environments.

We identified and isolated the impact of WAN environments by quantifying and comparing the performance of thin-client systems in both WAN and local-area network (LAN) environments. Because many thin-client systems are proprietary and closed-source, we employed a slow-motion benchmark [38] technique for obtaining our results, addressing some of the fundamental difficulties in previous studies of thin-client performance. Our results show that thin-client computing in a WAN environment can deliver acceptable performance over Internet2, even when client and server are located thousands of miles apart on opposite ends of the country. However, performance varies widely and not all approaches are suitable for this environment. We show that commonly used performance optimizations that work well for reducing the network bandwidth requirements of thin-client systems can degrade overall system performance due to latencies seen in WAN environments. Our results show that a simple pixel-based remote display approach can deliver superior performance compared to more complex thin-client systems that are currently used. We analyze the differences in the underlying mechanisms of various thin-client platforms and explain their impact on overall performance.

This paper is organized as follows. Section 2 details the experimental testbed and methodology we used for our study. Section 3 describes our measurements and performance results. Section 4 discusses related work. Finally, we present some concluding remarks and directions for future work.

2. EXPERIMENTAL DESIGN

The goal of our research is to compare thin-client systems to assess their basic display performance and their feasibility in WAN environments. To explore a range of different design approaches, we considered six popular thin-client platforms: Citrix MetaFrame 1.8 for Windows 2000, Windows 2000 Terminal Services, Tarantella Enterprise Express II for Linux, AT&T VNC v3.3.2 for Linux, Sun Ray I, and X11R6 on Linux. In this paper, we also refer to these platforms by their remote display protocols, which are Citrix ICA (Independent Computing Architecture), Microsoft RDP (Remote Desktop Protocol), Tarantella AIP (Adaptive Internet Protocol), VNC (Virtual Network Computing), Sun Ray, and X, respectively. As summarized in Table 1, these platforms span a range of differences in the encoding of display primitives, policies for updating the client display, algorithms for compressing screen updates, supported display color depth, and transport protocol used. To evaluate their performance, we designed an experimental Internet2 testbed and various experiments to exercise each of the thin-client platforms on single-user web-based and multimedia-oriented workloads. Section 2.1 introduces the non-invasive slow-motion measurement methodology we used to evaluate thin-client performance. Section 2.2 describes the experimental testbed we used. Section 2.3 discusses the mix of micro-benchmarks and application benchmarks used in our experiments.

2.1 Measurement Methodology

Because thin-client systems are designed and used very differently from traditional desktop systems, quantifying and measuring their performance effectively can be difficult. In traditional desktop systems, an application typically executes and displays its output on the same machine. In thin-client systems, an application executes on a server machine and sends its output over a network to be displayed on a client machine. The output display on the client may be completely decoupled from the application processing on the server such that an application runs as fast as possible on the server without regard to whether or not application output has been displayed on the client. Furthermore, display updates may be merged or even discarded in some systems to conserve network bandwidth. Since the server processes all application logic in thin-client systems, standard application benchmarks effectively measure only server performance and do not accurately reflect user perceived performance at the client. The problem is exacerbated by the fact that many thin-client systems, including those from Citrix, Microsoft, and Tarantella, are proprietary and closed-source, making it difficult to instrument them to obtain accurate, repeatable performance results.

To address these problems, we employed slow-motion benchmarking to evaluate thin client performance. This method employs two techniques to obtain accurate measurements: monitoring client-side network activity and using slow-motion versions of application benchmarks. We give a brief overview of this technique below. For a more in depth discussion, see [38]. We then extended this technique to compare relative performance across LAN and Internet2 network environments.

We monitored client-side network activity to obtain a measure of user-perceived performance based on latency. Since we could not directly peer into the black-box thin-client systems, our primary measurement technique was to use a packet monitor to capture resulting network traffic on the client-side. For example, to measure the latency of an operation from user input to client output, we could use the packet monitor to determine when the user input is first sent from client to server and when the screen update finished sending from server to client. The difference between these times could be used as a measure of latency. To accurately measure user-perceived thin-client performance, measurements must be performed at the client-side; server-side measurements of application performance are insufficient. For instance, a video application might deliver smooth playback on the server-side only to deliver poor video quality on the client-side due to network congestion. It must be noted that this measurement technique does not include the time from when the client receives a screen update from the network to the time the actual image is drawn to the screen. The measurement also does not include the time from when client input is made and the input is sent. Using VNC, one of the few open-source thin-client systems, we have verified that measurements using packet monitoring of slow-motion benchmarks are within five percent of internal client and server source code instrumentation [38]. We therefore assumed the client input and display processing times were negligible in our experiments.

We employed slow-motion versions of application benchmarks to provide a measure of user-perceived performance based on the visual quality of display updates. While monitoring network activity provides a measure of the latency of display updates, it does not provide a sufficient measure of the overall quality of the performance. To address this problem, we altered the benchmark applications used by introducing delays between the separate visual components of each benchmark, such as web pages or video frames, so that the display update for each component is fully completed on the client before the server begins processing the next display update. We monitored network traffic to make sure the delays were long enough to provide a clearly demarcated period between display updates where client-server communication drops to the idle level for that platform. We then process the results on a per-component basis to obtain the latency and data transferred for each visual component, and obtain overall results by taking the sum of these results. Section 2.3 describes in further detail how web and video application benchmarks were delayed for our experiments.

We compare relative performance across LAN and Internet2 network environments to isolate the impact of WAN environments on thin client performance, which can be quantified as the difference in performance between the two environments. Furthermore, this relative performance measure allows us to factor out effects of client processing time, which we cannot directly measure because of the proprietary nature of most of the thin-client systems. We assume that client processing time does not change in any significant way as a result of different network environments and we verified this assumption in the few open-source platforms tested.

Our combined measurement techniques provide three key benefits. First, the techniques ensure that display events reliably complete on the client so that packet captures from network monitoring provide an accurate measure of system performance. Ensuring that all clients display all visual components in the same sequence provides a common foundation for making comparisons among thin-client systems. Second, the techniques do not require any invasive modification of thin-client systems. As a result, we are able obtain our results without imposing any additional performance overhead on the systems measured. More importantly, the techniques make it possible for us to measure popular but proprietary thin-client systems, such as those from Citrix and Microsoft. Third, by comparing performance in LAN and WAN environments, we are able to isolate and analyze the impact of WAN effects on thin client performance.

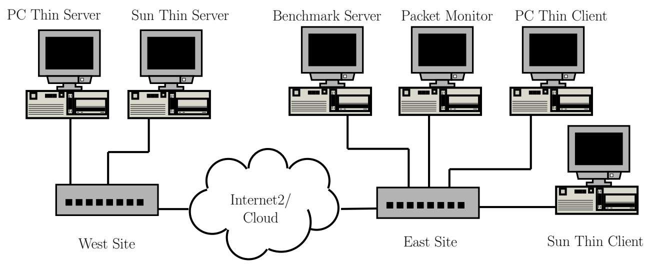

Figure 1: Experimental testbed

2.2 Experimental Testbed

Figure 1 shows our Internet2 testbed. The testbed consisted of two pairs of thin client/server systems, a packet monitor machine, and a web server used for testing web applications. The features of each system are summarized in Table 2. To ensure a level playing field, where possible we used the same hardware for all of our tests; the only change we made to our configuration was for testing the Sun Ray platform, which runs only on Sun machines. The machines were logically grouped into West and East sites separated by a network, with the thin servers located at the West site and the remaining three machines located at the East site. For the Internet2 experiments, we located the West and East sites on opposite coasts of the United States at Stanford University and Columbia University, respectively. We selected sites that were geographically far apart as a measure of the limits of using thin-client computing in wide-area environments.

The East site consisted of a PC thin client, a Sun Ray thin client, a packet monitor machine, and a benchmark server. The packet monitor machine was dedicated to running Etherpeek 4 [13], a software packet monitor that timestamps and records all packet traffic visible by the machine. Except for the Sun Ray thin client, all other East site machines were Micron Client Pro PCs, each with a 450 Mhz Pentium II CPU, 128 MB RAM, and 14.6 GB disk. The Sun Ray client was considerably less powerful than the PC client, with only a 100 Mhz uSPARC CPU and 8 MB of RAM. The West site consisted of a PC server and a Sun server. The PC server was a Hi-Tech USA PC with dual 500 Mhz Pentium III CPUs, 160 MB RAM, and 22 GB disk. The Sun server was an Ultra-10 Creator 3D with a 333 Mhz UltraSPARC IIi, 384 MB RAM, and 9 GB hard disk. All machines had 10/100BaseT NICs. As discussed in Section 3.2, the slower Sun client and server hardware did not affect the lessons derived from our experiments.

For the Internet2 experiments, the East and West sites were connected by the Abilene Internet 2 backbone, an OC-48 operating at 2.5 Gbps, typically very lightly loaded with line utilization usually below 10% on any given link of the backbone [1]. A total of 14 hops separate the machines at the East site from those at the West site. The East site is three hops away from a network provider that connects to the Abilene Internet2 backbone via an OC-12 (622 Mbps), with all links being OC-3 (155 Mbps) or greater. The West site is three hops away from a GigaPOP that connects to the Abilene Internet2 backbone via an OC-12, with all links being OC-3 (155 Mbps) or greater. The machines at each site are connected to a 100Base-T hub which is uplinked to the respective core networks via 100Base-T full-duplex switches. Since the minimum available bandwidth along any edge of the network was 100 Mbps, the theoretical maximum bandwidth of our connection was 100 Mbps.

Role / Model	**Hardware**	**OS / Window System**	**Software**
PC Thin Client Micron Client Pro (East site)	450 MHz Intel PII 128 MB RAM 14.6 GB Disk 10/100BaseT NIC	MS Win 2000 Professional Caldera OpenLinux 2.4, XFree86 3.3.6, KDE 1.1.2	Citrix ICA Win32 Client MS RDP5 Client VNC Win32 3.3.3r7 Client Tarantella Win32 Client Netscape Communicator 4.72
Sun Thin Client Sun Ray I (East site)	100 MHz Sun uSPARC IIep 8 MB RAM 10/100BaseT NIC	Sun Ray OS	N/A
Packet Monitor Micron Client Pro (East site)	450 MHz Intel PII 128 MB RAM 14.6 GB Disk 10/100BaseT NIC	MS Win 2000 Professional	WildPackets' Etherpeek 4
Benchmark Server Micron Client Pro (East site)	450 MHz Intel PII 128 MB RAM 14.6 GB Disk 10/100BaseT NIC	MS Win NT 4.0 Server SP6a	Ziff-Davis i-Bench 1.5 MS Internet Information Server
PC Thin Server Hi-Tech USA (West site)	2 500 MHz Intel PIII 160 MB RAM 22 GB Disk 10/100BaseT NIC	MS Win 2000 Advanced Server Caldera OpenLinux 2.4, XFree86 3.3.6, KDE 1.1.2	Citrix MetaFrame 1.8 MS Win 2000 Terminal Services AT&T VNC 3.3.3r7 for Win32 Tarantella Express AT&T VNC 3.3.3r2 for Linux Netscape Communicator 4.72
Sun Thin Server Sun Ultra-10 Creator 3D (West site)	333 MHz UltraSPARC IIi 384 MB RAM 9 GB Disk 2 10/100BaseT NICs	Sun Solaris 7 Generic 106541-08 OpenWindows 3.6.1, CDE 1.3.5	Sun Ray Server 1.2_10.d Beta Netscape Communicator 4.72
Network Simulator Micron Client Pro (simulator testbed)	450 MHz Intel PII 128 MB RAM 14.6 GB Disk 2 10/100BaseT NICs	MS Win NT 4.0 Server SP6a	Shunra Software The Cloud 1.1

Table 2: Testbed machine configurations

Based on our own measurements and a month of sampled data obtained by the National Laboratory for Applied Network Research (NLANR), ping results have shown the mean round trip time (RTT) latency to be approximately 66.35 ms with a standard deviation of 4.52 ms and a minimum RTT of 64 ms [2, 3]. The measured percentage packet loss over this Internet2 connection was less than 0.05%.

Because all of the thin-client systems tested, except for Sun Ray, used TCP as the underlying network transport protocol, we were careful to consider the impact of TCP window sizing on performance. TCP windows should be adjusted to at least the bandwidth delay product size to maximize bandwidth utilization [19]. Otherwise, the effective bandwidth available can be severely limited because the largest amount of data that can be in transit without acknowledgement is the TCP window size being used. When at the default window size of 16 KB under Windows [11] and at our average RTT latency of 66 ms, there is a maximum theoretical bandwidth availability of only 1.9 Mbps. With an RTT latency of 66 ms, the optimal TCP window size is 825 KB in order to take full advantage of the 100 Mbps Internet2 network bandwidth capacity available. Because of this, we decided to test with the operating system defaults as well as a high network latency optimized large TCP window setting. To make things simple and ensure that the window size was large enough even if the network latency increased, a large TCP window size of 1 MB was used. After making this optimization, iperf [33] was used to determine that the actual bandwidth available to us over Internet2 was approximately 45 Mbps. For all of the experiments on TCP-based thin-client systems, we also ensured that the TCP sliding window reached the TCP window size before running the application benchmarks.

To verify our results in a more controlled network environment and to provide a basis for comparison, we also constructed a local isolated testbed for comparison purposes, also shown in Figure 1. The local testbed structure was similar to the Internet2 testbed, except that a network simulator was used instead of Internet2 for the network connection between the East and West site. The network simulator used was a Micron Client Pro PC with two 100BaseT NICs running The Cloud [7], a network simulator that has the ability to adjust the bandwidth, latency, and packet loss rate between the East and West sites. We used the local testbed in two ways. First, we used the local testbed network as a 100 Mbps low latency LAN testbed environment to allow us to compare thin client performance over Internet2 versus a LAN environment. Second, we adjusted the local testbed network characteristics to match the measured characteristics of the Internet2 testbed so that we could verify our Internet2 testbed measurements in a more controlled network environment. All platforms were evaluated in both Internet2 and the simulated Internet2 testbed except for Sun Ray, which was only evaluated in the simulated Internet2 testbed due to the difficulty of configuring its dynamic authentication over Internet2. There was no significant difference in our measurements in this simulated Internet2 testbed compared to our measurements during periods of light network load over Internet2. We therefore assume that Sun Ray would also have no significant performance difference between the two testing environments. We also determined that the amount of packet loss observed over Internet2 was low enough to be considered negligible in our experiments by comparing measurements with both 0% loss and 1% loss on our simulated Internet2 testbed. In all tests performed, there was no significant difference between the two cases.

To minimize application environment differences, we used common thin-client configuration options and common applications across all platforms whenever possible. Where it was not possible to configure all the platforms in the same way, we generally used default settings for the platforms in question. In particular, unless otherwise stated, the video resolution of the client was set to 1024x768 resolution with 8-bit color, compression and memory caching were left on for those platforms that used it, and disk caching was turned off by default in those platforms that supported it. A study on the impact of caching on thin-client systems in WAN environments is ongoing but is beyond the scope of this paper. For each thin-client system, we used the server operating system which delivered the best performance for the given system; Terminal Services only runs on Windows, Citrix ran best on Windows, Tarantella, VNC, and X ran best on UNIX/Linux, and Sun Ray only runs on Solaris.

2.3 Application Benchmarks

To measure the performance of the thin-client platforms, we used three application benchmarks: a latency benchmark for measuring response time, a web benchmark for measuring web browsing performance, and a video benchmark for measuring video playback performance. The latency benchmark was used as a micro-benchmark to measure simple operations while the web and video benchmarks were used to provide a more realistic measure of real application performance. We describe each of these benchmarks in further detail below. In particular, the web and video benchmarks were used with the slow-motion benchmarking technique described in Section 2.1 to measure thin client performance effectively.

2.3.1 Latency Benchmark

The latency benchmark used was a small Java applet that permitted us to run five separate tests:

- Letter: a character typing operation that took a single keystroke as input and responded by displaying a 12-point capital letter 'A' in sans serif font.
- Scroll: a text scrolling operation that involved scrolling down a page containing 450 words in 58 lines in 12-point sans serif font, with 14 of the lines displayed in a 320x240 pixel area at any one time.
- Fill: a screen filling operation in which the system would respond to a mouse click by filling a 320x240 pixel area with the color red.
- Red Bitmap: a bitmap download operation in which the system would respond to a mouse click by displaying a 1.78 KB JPEG red bitmap at 320x240 pixels in size.
- Image: an image download operation in which the system would respond to a mouse click by displaying a 15.5 KB JPEG image at 320x240 pixels in size.

For our experiments, we measured the latency of each test from the time of user input until the time that the client receives the last screen update from the server. This time is measured using packet trace data collected by the packet monitor. The time is calculated as the difference between the timestamp of the first client-to-server packet and the timestamp of the last server-to-client packet for the respective test.

2.3.2 Web Benchmark

The web benchmark we used was based on the Web Text Page Load test from the Ziff-Davis i-Bench benchmark suite [14]. The original i-Bench web benchmark is a JavaScript-controlled load of a sequence of 54 web pages from the web benchmark server. Normally, as each page downloads, a small script contained in each page starts off the subsequent download. The pages contain both text and bitmap graphics, with some pages containing more text while others contain more graphics. The JavaScript cycles through the page loads twice, resulting in a total of 108 web pages being downloaded during this test. When the benchmark is run from a thin client, the thin server would execute the JavaScript that sequentially requests the test pages from the i-Bench server and relay the display information to the thin client. For the web benchmark used in our tests, we modified the original i-Bench benchmark for slow-motion benchmarking by introducing delays of several seconds between pages using the JavaScript, sufficient in each case to ensure that the thin client received and displayed each page completely and there was no temporal overlap in transferring the data belonging to two consecutive pages. We used the packet monitor to record the packet traffic for each page, and then used the timestamps of the first and last packet associated with each page to determine the download time for each page. We used Netscape Navigator 4.72 to execute the web benchmark, as it is available on all the platforms in question. The browser's memory cache and disk cache were enabled but cleared before each test run. In all cases, the Netscape browser window was 1024x768 in size, so the region being updated was the same on each system.

2.3.3 Video Benchmark

The video benchmark used processes and displays an MPEG1 video file containing a mix of news and entertainment programming. The video is a 34.75 second clip that consists of 834 352x240 pixel frames with an ideal frame rate of 24 frames/sec. The total size of the video file is 5.11 MB. The thin server executed the video playback program to decode the MPEG1 video then relayed the resulting display to the thin client. In systems which have a lazy screen update mechanism, acting as frame buffer scrapers, frames that are drawn to the virtual framebuffer on the server between screen update requests are simply not relayed to the client. In systems which have an eager update mechanism where the display updates are encoded and sent at the time the server window system command occurs, the video application measures the time differential between the time the frame update was issued and completed. If the time differential is too great, the application then drops the intermediate frames to compensate. Because of this behavior, we measured video performance using slow-motion benchmarking by monitoring resulting packet traffic at two playback rates, 1 frames/second (fps) and 24 fps. Although no user would want to play video at 1 fps, we took the measurement at that frame rate to ensure all data packet from the thin server to the client were recorded in order to establish the reference data size transferred from the thin server to the client that corresponds to a "perfect" playback. To measure the normal 24 fps playback performance and video quality, we monitored the packet traffic delivered to the thin client at the normal playback rate and compared the total data transferred to the reference data size. This ratio multiplied by 24 fps would yield the real effective frame rate of the playback [38]. For the video benchmark, we used two different players capable of playing MPEG1 files. We used Microsoft Windows Media Player version 6.4.09.1109 for the Windows-based thin clients and MpegTV version 1.1 for the Unix-based thin clients. Both players were used with non-video components minimized so that the appearance of the video application was similar across all platforms.

3. EXPERIMENTAL RESULTS

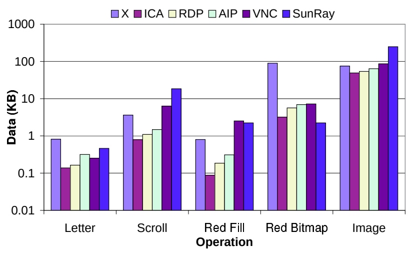

Figure 2: Data Transfer of Operations

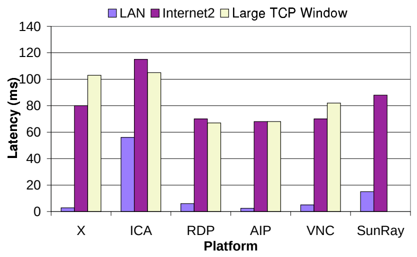

Figure 3: Letter Latency

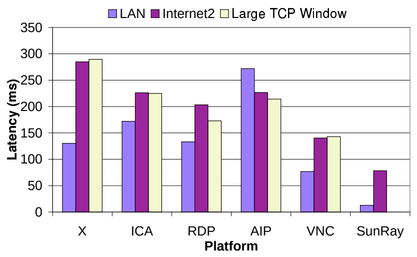

Figure 4: Scroll Latency

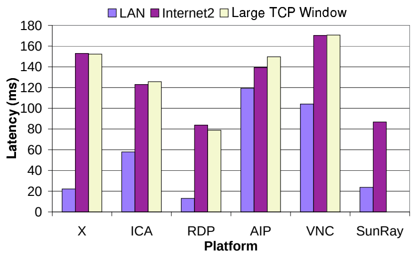

Figure 5: Fill Latency

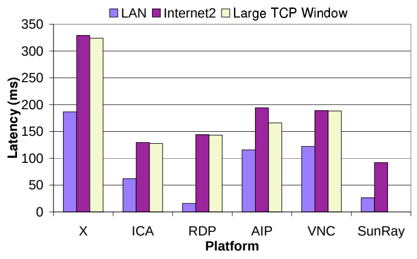

Figure 6: Red Bitmap Latency

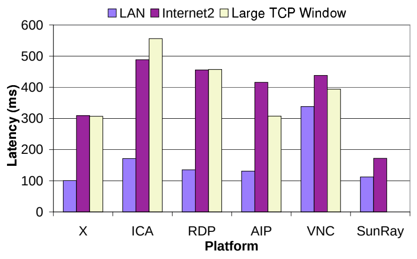

Figure 7: Image Latency

We ran the three benchmarks on each of the six thin-client platforms and measured their resulting performance in both the Internet2 testbed and the local LAN testbed environments. The results reported here for the local testbed were with the network simulator configured to represent a 100BaseT LAN network environment. Results are shown both in terms of the amount of data transferred and the respective latencies to indicate both the bandwidth efficiency of the thin-client systems as well overall user-perceived performance. For thin-client systems based on TCP/IP, we report results over Internet2 for both the default TCP window sizes and the large 1 MB window sizes used. Section 3.1 presents an overview of the measurements obtained and provides some metrics of performance for each application benchmark. These measurements provide the first quantitative performance comparisons of thin-client systems in WAN environments. Section 3.2 discusses the implications for how thin-client systems should be designed for WAN environments.

3.1 Measurements

Figures 2 to 7 show the results of running the latency benchmark on each of the six thin-client systems. The figures refer to the thin-client systems based on their remote display protocols. Figure 2 shows the amount of data transferred for each operation on each thin-client system over Internet2. The data transferred for the LAN and Internet2 with large TCP window sizes was similar for almost all platforms and is not shown due to space constraints. Tarantella AIP was an exception and is discussed further in Section 3.2.4. Figures 3 to 7 show the latency of the letter, scroll, fill, red bitmap, and image operations on each system, respectively. Generally, for simple tasks such as typing, cursor motion, or mouse selection, system response should be less than the 50-150 ms threshold of human perception to keep users from noticing any delay [30]. Figures 3 to 7 show that several of the systems performed better than the 150 ms threshold for many of the operations. Sun Ray stands out as having less than 100 ms latency for both LAN and Internet2 environments for almost all operations. Only the image operation took a little longer than 150 ms, and the 150 ms threshold used for simple tasks arguably does not apply for such a complex operation.

Figures 8 and 9 show the results of running the web benchmark on each of the six thin-client systems. Figure 8 shows the average amount of data transferred per web page over a LAN, Internet2, and Internet2 with 1 MB TCP window sizes. The amount of data transferred for each platform was approximately the same in each of the network conditions tested. Tarantella AIP transferred slightly less data over Internet2 compared to the LAN. However, the difference was less than ten percent and is attributable to the adaptive compression capabilities of the platform, as discussed in Section 3.2.4. Figure 9 shows the average latency per web page. Usability studies have shown that web pages should take less than one second to download for the user to experience an uninterrupted browsing process [23]. Our results show that while VNC achieved the best Internet2 web performance, most of the platforms performed well over Internet2, with each web page taking less than a second on average to download and display. Only X showed poor performance over Internet2, taking over six seconds on average to display each web page.

Figures 10 to 12 show the results of running the video benchmark on each of the six thin-client systems over a LAN, Internet2, and Internet2 with 1 MB TCP window sizes. Figure 10 shows the amount of data transferred during normal video playback at 24 fps. Unlike the latency and web benchmark results, there are substantial differences in the amount of data each platform transferred among the different network conditions tested. Figure 10 also shows the amount of data transferred during video playback when the playback rate was set to 1 fps. At 1 fps, all of the video frames were rendered completely on the client and the data transferred for each platform was similar over LAN, Internet2, and Internet2 with 1 MB TCP windows. Figure 11 shows the video playback time on each system. Except for X, there was relatively little variation in playback time across different network environments. Figures 10 and 11 taken together indicate that when the thin-client systems cannot deliver the video at the desired playback rate, most of them simply discard data rather than slowing down the video. Figure 12 shows the quality of video delivered on each thin-client system, calculated as described in Section 2.3.3 by comparing the measured results at 24 fps versus the slowed down playback at 1 fps. Unlike the web

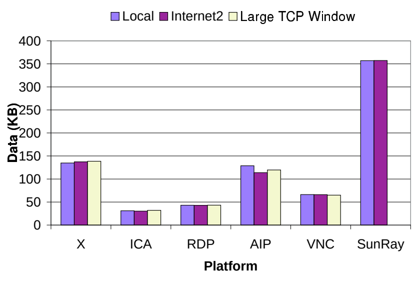

Figure 8: Web Data Transfer

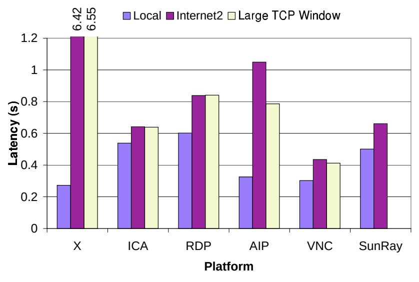

Figure 9: Web Latency

benchmark in which most of the thin-client systems delivered reasonable performance, Figure 12 shows that most of the thin-client systems performed poorly on the video benchmark over both LAN and Internet2 environments. Only Sun Ray's performance was reasonable over Internet2, delivering roughly 70% video quality. The video quality achieved on all of the other platforms was below 35% and generally not usable.

For most of the TCP-based thin-client platforms, there was not a significant performance difference when running the benchmarks over Internet2 with default TCP window sizes versus the 1 MB TCP window sizes. Figures 9 and 12 show that Tarantella AIP and VNC performed better with the larger TCP window sizes on the web and video benchmarks, respectively. The more pronounced performance difference occurred with the video benchmark. The use of larger TCP window sizes made a bigger difference there due to the higher data bandwidth requirements of video. In some cases, using larger window sizes resulted in slightly higher overhead. When using increased window sizes, RFC1323 options must be used which increases the sequence number field from 2 bytes to 4 bytes per packet and adds an additional window scaling field. These additional fields may add some overhead to the processing of each packet, the effect of which is exaggerated when the payload of the packet is small. These additional fields also resulted in slightly more data being transferred when using large window sizes, but the difference was only a few percent in all cases.

3.2 Interpretation of Results

The measurements presented in Section 3.1 show that using thin-client computing in a WAN environment can deliver acceptable performance over Internet2, even when client and server are located thousands of miles apart on opposite ends of the country. In particular, Sun Ray delivered excellent performance on all of the application benchmarks measured. However, performance varies widely among thin-client platforms and not all platforms are suitable for this environment. Due to space constraints, it is not possible to describe all results and their implications in detail in this paper. However, we discuss five principles that should serve as guidelines in designing thin-client systems for supporting wide-area computing services: optimize latency over bandwidth, partition client/server functionality to minimize synchronization, use simpler display primitives for speed, compress display updates, and push display updates.

Our results are based on measurements that do not account for the impact of loss and congestion found in commodity WAN environments. We have done preliminary testing in the simulated Internet2 testbed with a 10% random packet loss, which results in low link utilization [17]. These tests suggest that the principles discussed here are likely to continue to hold true under loss and congestion, but further study is required and is the beyond the scope of this paper.

3.2.1 *Optimize Latency vs Bandwidth*

Although thin-client computing vendors often tout the bandwidth efficiency of their platforms, our measurements show that the bandwidth efficiency of a thin-client system is not a good predictor of performance over Internet2. Figures 2, 8, and 10 show that Citrix ICA and Microsoft RDP usually transferred less data overall for each benchmark compared to the other systems while Sun Ray typically transferred the most amount of data overall for each benchmark. However, in terms of user-perceived performance, our measurements show that overall Sun Ray significantly outperformed both ICA and RDP over Internet2 for both the latency and video benchmarks and was comparable for the web benchmark. For the latency benchmark, Figures 3 to 7 show that ICA and RDP have response times that balloon to over 400 ms over Internet2 while Sun Ray response times remain near 150 ms or less. For the video benchmark, Figure 12 shows that Sun Ray delivered video quality that was more than four times better than either ICA or RDP. For the web benchmark, the web browsing latency for Sun Ray was comparable to ICA and better than RDP despite sending almost an order of magnitude more data. Furthermore, while ICA and RDP sent the least amount of data per page, 30 KB and 41 KB respectively, VNC had the lowest latency over Internet2 with an average page latency of 410 ms, 50 percent faster than ICA and twice as fast as RDP.

Our measurements show that bandwidth availability in LAN or Internet2 environments was not the main performance limitation for both the latency and web benchmarks, assuming appropriately sized TCP windows. For the latency benchmark, the most bandwidth consumed for any of the operations was 11 Mbps for the image operation on Sun Ray. For the web benchmark, no platform consumed more than 5 Mbps on average. Only in the video benchmark did one of the platforms, Sun Ray, approach the limits of bandwidth available, consuming roughly 33 Mbps. However, despite using the most bandwidth for the video benchmark, Sun Ray delivered by far the best video performance over Internet2.

Instead of network bandwidth being the primary bottleneck, our measurements comparing thin client performance over Internet2 versus the LAN show that network latency had a significant impact on thin client performance. For the latency benchmark, Figures 3 to 7 show that the latency of operations over Internet2 for almost all of the thin-client systems were roughly 65 ms or more longer than the results for the same operation over the LAN testbed. AIP was an exception to this which we discuss further in Section 3.2.4. The reason for the added latency is because each operation requires the client to send input to the server and the server to reply with the display update, which entails at least one round trip across the network. Since the RTT for Internet2 is roughly 65 ms longer than

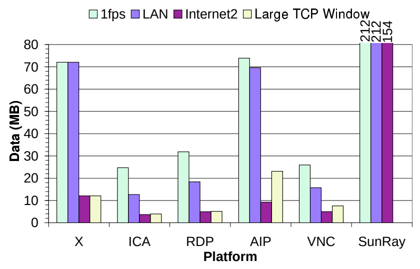

Figure 10: Video Data Transfer

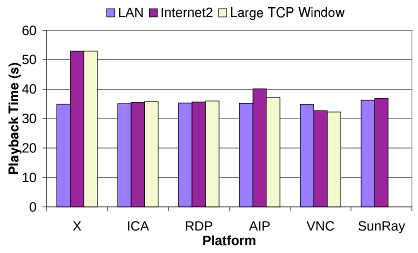

Figure 11: Video Playback Time

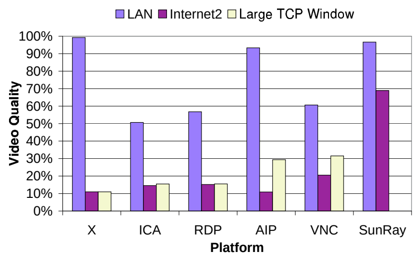

Figure 12: Video Quality

the LAN, it should be expected that the operations would take 65 ms longer over Internet2 versus the LAN. What was not expected is that only Sun Ray and VNC took no more than one RTT longer for each operation over Internet2 versus the LAN. All of the other systems incurred more than one RTT of additional latency over Internet2 versus the LAN for some of the operations. This implies that X, ICA, RDP, and AIP in some cases required multiple round trip times to complete a simple operation, making them less tolerant of the increased network latencies found in WAN environments.

For the web benchmark, our measurements also show the impact of network latency on thin-client performance. Figure 9 shows that the average per web page download latency over Internet2 was roughly 65 ms or more longer than the same latency over a LAN for a given platform. In our Internet2 testbed shown in Figure 1, the thin-client server was located across the Internet2 network from the web server. Since the thin-client server is local to the web server in the LAN, we can expect an extra Internet2 RTT delay in downloading a web page over Internet2 versus the LAN because of extra delay between getting the container HTML page to the thin-client server and the subsequent retrieval of the images. In addition, the Netscape status widgets also induce an additional Internet2 RTT over Internet2. Since the web browser caches the web pages, the second run through the 54 web pages does not cause the browser to re-request the web pages from the web server. As a result, there is at best a two RTT additional delay for the first 54 pages only, or an average of one RTT additional delay per page. Only Sun Ray, VNC, and ICA incurred this minimum increased delay for the Internet2 versus LAN web benchmark results. While these three platforms performed the best over Internet2, the amount of data transferred for these systems varied from the smallest to the largest amount of data transfer for all of the platforms, showing no correlation with bandwidth consumption.

Our results demonstrate the importance of designing wide-area thin-client systems with a focus on tolerating network latency as opposed to just minimizing network bandwidth. As network technologies improve and high-bandwidth Internet access becomes a commodity, the fundamental physical limits of propagation delay lead us to believe that the ability of thin-client systems to tolerate latency will be an increasingly dominant factor in determining their performance.

3.2.2 *Partition Client/Server to Minimize Synchronization*

The design of a thin-client system requires that the functionality of the system be partitioned in some manner between the client and the server. An important partitioning issue is to what extent is the client's graphical user interface functionality supported on the client versus the server. For instance, Sun Ray and VNC do not support any windowing functions on the client but instead maintain all thin-client window system state at the server. On the other hand, X maintains significant window system state at the client to allow the client to locally manage window positioning, colormap information, font libraries, etc. X stands apart from the other thin-client systems in the degree in which it uses local client window system state, which also makes the client-side of an X system more heavyweight than those of other thin-client systems. Compared with Sun Ray and VNC, the partitioning of functionality between client and server in X potentially allows X to perform more operations locally at the client, but may require more coordination between the client and server for display updates sent from the server.

While the X approach performs quite well over a LAN, overall it performs far worse over Internet2 than all of the other thin-client systems. For the latency benchmark, Figures 3 to 7 show that X incurred two to three Internet2 RTT of additional latency over Internet2 versus a LAN for all operations except the letter operation. This is in stark contrast with the other platforms which for many of the operations only suffered the minimum of one RTT of additional latency over Internet2 versus a LAN. Given the Internet2 RTT of 66 ms, X has over 130 ms of additional latency for operations over Internet2, resulting in slower response time that is very noticeable to the user. For the web benchmark, Figure 9 shows that X provides the best performance over a LAN, but the absolute worst performance over Internet2. X on average took well over 6 seconds per web page over Internet2. For the video benchmark, Figure 12 shows again that X provides the best video quality over a LAN but only 11% video quality over Internet2, the worst video quality of all the platforms assuming large TCP windows. Our results suggest that this partitioning of client/server display functionality in X requires a higher degree of synchronous coordination between client and server than systems which do not employ as much local client window system functionality. The better performance results with thin-client systems such as Sun Ray surprisingly suggest that minimizing the amount of local client window system functionality can result in better overall performance in WAN environments.

The video benchmark perhaps most clearly shows the problems that result from synchronization of the client and server with the X protocol. In comparing the amount of data X transferred over Internet2 versus the LAN as shown in Figure 10, we can see that the primary problem is that most of the video data does not get sent to the client. The X display command does not complete until the client actually receives the video frame from the server and returns an acknowledgement, incurring a cross-country round trip delay for each video frame displayed. Because the application realizes that it cannot display the next video frame on time, it skips ahead in the video stream until it finds a video frame that can be displayed on time. The intermediate video frames are dropped and not displayed, resulting in degraded video quality, effectively showing only one out of every 10 frames. A secondary factor for X's poor video quality is that it takes much longer to playback the video stream. The 17 seconds of extra delay is due to the level of synchronized coordination between X client and server in allocating the color map used throughout the playback of the video. We verified that this was not just an artifact of MpegTV by testing the popular Berkeley Multimedia Research Center's mpeg_play, which also exhibited the same extra delay.

Because X is an application-level protocol, its performance depends heavily on what X primitives an application is programmed to use. X does have the have the ability to support client/server display functions that are more decoupled between client and server. However, our experiments with widely-used commercial X applications such as Netscape show that it is not uncommon to find a high proportion of synchronous display functions used. In particular, whenever there is a return value that reports the status of a request, the operation must be completed synchronously and the application waits for the return value. Unfortunately in Netscape, all of the routines that draw the toolbar and the page load status bar create a significant number of GetWindowAttributes and GetGeometry requests, which are both synchronous functions. If widely-used commercial X applications can so easily have performance problems in WAN environments, it seems clear that the X system itself is at least partially to blame even if X primitives may exist that allow more decoupled client-server interactions.

We can quantify to some extent the degree of synchronization in a system's display protocol by the amount of extra delay experienced running each system over Internet2 versus a LAN. For the latency benchmark, Sun Ray and VNC incur the minimum delay as discussed in Section 3.2.1. ICA provides the next best performance, incurring the minimum delay except for the image operation. RDP does a little worse, requiring two extra RTT of delay on both the red bitmap and image operations. X does the worst. From inspecting the packet captures in the latency benchmark, it appears that ICA and RDP perform some sort of synchronized operation after approximately every 8 KB of data being sent. When this occurs, the protocols each wait a full RTT before continuing with the remaining data transfer. This synchronized execution also limits the utility of using larger TCP window sizes, as is evident by the lack of improvement in performance when using larger TCP window sizes versus default TCP window sizes. For the web benchmark, Sun Ray, VNC and ICA incur the minimum extra delay over Internet2 as discussed in Section 3.2.1. They are followed in best-to-worst relative performance order by RDP, AIP, and X. Unfortunately, due to the proprietary nature of ICA, RDP, and AIP, it was not possible to examine in detail the mechanisms behind the synchronization of the protocols.

Overall, the degree of synchronization in the display protocol has a much more significant impact over Internet2 than bandwidth efficiency. Our results demonstrate that to optimize performance for the larger latencies in WAN environments, the functionality in a thin-client system should be carefully partitioned between the client and server to minimize synchronization between client and server. If the client and server need to send messages back and forth several times to perform an operation, the much higher round trip latencies over Internet2 will result in significant increases in latency for the given operation.

3.2.3 *Use Simpler Display Encoding Primitives*

Different thin-client systems use different display primitives for encoding display updates that are sent from the server to the client. Four types of display encoding primitives are high-level graphics, low-level graphics, 2D draw primitives, and raw pixels. Higher-level display encodings are generally considered to be more bandwidth efficient, but require more complexity on the client and may be less platform-independent. For instance, graphics primitives such as fonts require the thin-client system to separate fonts from images while using pixel primitives enable the system to view all updates as just regions of pixels without any semantic knowledge of the display content. X takes a high-level graphics encoding approach and supports a rich set of graphics primitives in its protocol. ICA, RDP, and AIP are based on lower-level graphics primitives that include support for fonts, icons, drawing commands as well as images. Sun Ray and VNC employ 2D draw primitives such as fills for filling a screen region with a single color or a two-color bitmap for common text-based windows. VNC can instead be configured to use raw pixels only, but none of the systems we considered used raw pixels by default.

Our results show that higher-level display encodings are not necessarily more bandwidth efficient than lower-level primitives. For the latency benchmark, Figure 2 shows that the low-level graphics encodings such as ICA, RDP, and AIP generally required less data transfer than the pixel-based approaches such as VNC and Sun Ray, but the high-level X encoding format required the highest data transfer on two of the five latency operations. For the web benchmark, Figure 8 shows that while the higher-level encoding formats used by ICA and RDP require less data transfer than the lower-level pixel-based encoding formats used by VNC and Sun Ray, VNC sends less data than either X or AIP, which also use higher-level encoding formats. Furthermore, while Sun Ray transfers significantly more data than the other platforms, this is largely because its remote display protocol encodes pixel values in 24-bit color [28]. If we normalize the amount of data transferred by the number of bits used for pixel color in the protocol, the amount of data sent using the 24-bit color Sun Ray encoding would be three times less than that shown in Figure 8, since all the other platforms used 8-bit color for the experiments. The normalized Sun Ray data transfer would be about 110 KB per web page, less than both X and AIP but still more than ICA. However, ICA achieves some of its bandwidth efficiency by using compression. When we turned off compression in ICA to reveal the performance of its basic display encoding on the web benchmark, the data transfer requirement for ICA ballooned to about 100 KB per web page, only 10 percent less than Sun Ray.

A key reason why the higher-level display encoding primitives are often no better if not worse than the lower-level display encoding primitives is that many of these encodings were optimized for text-based displays. Much of the complexity of the higher-level encoding formats used by X, ICA, RDP, and AIP relates to keeping track of text-based primitives. But relative to images, graphics, and video, text generally does not require much bandwidth to begin with. Even for the web benchmark which consisted of mostly text-based web pages, text-oriented display accounts for much less than half of the data in the original HTML pages. Figure 10 shows that for the video benchmark which involves no text-oriented display, the higher-level encoding formats are not more bandwidth efficient than the lower-level formats. If we again normalize for the number of bits used for pixel color in the protocol, we see that X, AIP, and Sun Ray all require roughly the same amount of data transfer. Similarly, ICA, RDP, and VNC all require roughly the same amount of data transfer. ICA, RDP, and VNC require less data transfer than the other platforms simply because they use compression, as discussed in Section 3.2.4. As applications become more multimedia-oriented and bandwidth increases, the efficiency with which an encoding supports graphics and images is more important and additional complexity for text may in fact reduce performance.

More importantly, our measurements indicate that simpler lower-level display primitives as used by Sun Ray and VNC resulted in better overall user-perceived performance than higher-level display encoding primitives. Our results suggest that the higher-level primitives used in ICA, RDP, AIP, and X have higher latencies over Internet2 that may be due to their added complexity. For both the web and video benchmarks, Sun Ray and VNC outperformed all of the other higher-level encoding platforms. Figure 9 shows that Sun Ray and VNC had the lowest average web page latencies, with VNC being 50 percent better than any of the higher-level encoding platforms. Figure 12 shows that Sun Ray had the best video quality followed by VNC, with Sun Ray being more than two times better than any of the higher-level encoding platforms. While VNC's performance benefits substantially from compression as discussed in Section 3.2.4, Sun Ray's good performance is simply due to a good balance between computing and communication costs in its display encoding format. Note that the good performance results for Sun Ray were achieved despite using slower client and server hardware as compared to the other thin-client systems. When network bandwidth is sufficient and network latency is the primary issue, the simpler pixel-based encoding approaches provide better overall performance.

3.2.4 *Compress Display Updates*

As summarized in Table 1, many of the thin-client systems employ low-level compression techniques such as run-length encoding (RLE) and Lempel-Ziv Welch (LZW) compression to reduce the data size of display updates. For our experiments, compression was by default enabled on all of the thin-client systems that supported it. ICA, AIP, and VNC all provide a simple user option to enable or disable compression. To evaluate the impact of compression, we also ran the same benchmarks on these three thin-client systems with compression explicitly disabled and measured the resulting performance. As expected, all three platforms transferred less data on all of the benchmarks with compression enabled, though compression was least effective with the video benchmark. Furthermore, all three platforms performed better overall on the benchmarks with compression enabled as opposed to without it.

We identified three reasons why enabling compression improved performance. First, some of the thin-client systems, particularly VNC, were bandwidth limited when compression was disabled when default TCP window sizes were used. Enabling compression reduced the amount of data transferred and removed this bandwidth limitation. Second, two of the thin-client systems, ICA and RDP, require some synchronization between client and server after approximately every 8 KB of display update data that is sent, as discussed in Section 3.2.2. Enabling compression reduced the amount of data transferred and therefore reduced the frequency at which this synchronization occurred, thereby improving performance for WAN environments. Third, some of the thin-client systems may employ different client rendering functions depending upon whether compression is enabled. We discovered that when hextile compression is enabled, the VNC client rendering function renders blocks of pixels at one time. When compression is not used, the rendering function renders pixels individually, one at a time. Our measurements on the web and video benchmark showed that the rendering function applied when compression was used was 6 to 24 times faster per pixel displayed. As discussed further in Section 3.2.5, because the VNC server waits until the client has completely rendered the last display update before sending the next display update, the shorter client rendering times with compression enabled result in better performance.

Our results also show that low-level compression applied to a simple pixel-based display encoding as used in VNC can perform surprisingly well. In particular, the web benchmark results show that effective compression can compensate for a less efficient display encoding and dramatically reduce the amount of data that needs to be transferred without incurring significant additional overhead. This is most apparent from the data transfer and latency measurements for VNC. Figure 8 shows that VNC requires about 50% less data transfer than the higher-level X and AIP approaches, neither of which employed much if any compression over the network conditions considered for the web benchmark. Furthermore, Figure 9 shows that VNC had the lowest latency over Internet2 of any of the thin-client platforms for the web benchmark. The simple design of combining a low-level compression method with a simple pixel-based encoding provided very good performance on the web benchmark.

Because thin-client systems may operate in different network environments, adaptive compression mechanisms have been proposed to optimize the performance of these systems. AIP uses such an adaptive compression mechanism to turn on increasingly efficient compression algorithms as the available network bandwidth decreases. However, this adaption mechanism in some cases results in worse performance than expected. For instance, Figure 4 shows that AIP surprisingly has lower latency on the scroll operation over Internet2 than a LAN. The amount of data transferred over the LAN for this operation is many times larger than that which is transferred over Internet2, transferring approximately 92KB and 2KB respectively. The reason for this is because AIP adaptively disabled compression over the LAN but enabled compression over Internet2. When we manually enabled compression over the LAN, the scroll operation performance over the LAN was better than Internet2. Overall, our experimental results show that simple low-level compression can be used effectively to improve the performance of thin-client systems.

3.2.5 *Push Display Updates Eagerly*

The policy used to determine when display updates are sent from the server to the client is an important issue that does not receive the attention it deserves; when the display update is sent can be as important as what is sent. Two important display update policy issues are eager versus lazy display updates, and server-push versus client-pull models.

The first display update policy issue is whether display updates are sent eagerly with the server window system graphics commands or lazily as a framebuffer scraper. In the eager case, the display update is encoded and sent at the time the server window system command occurs. X and Sun Ray both do eager updates. In the lazy case, the window system command is queued in an intermediate representation, such as keeping track of regions of pixels that have been modified. Old modifications that are overwritten by newer modifications are discarded. Screen updates are then sent at regular intervals depending on available bandwidth, with only the latest modifications encoded and sent to the client. VNC, ICA, and RDP all perform lazy updates [20].

While lazy update mechanisms can be used to merge multiple display updates at the server for bandwidth efficiency, our measurements indicate that these mechanisms are often incompatible with the needs of multimedia applications such as video. For the video benchmark, Figure 12 shows that even over a LAN, all of the platforms that used lazy display updates delivered much worse quality video than those that used eager display updates. The problem that occurs in platform such as ICA and RDP is that the rate of their lazy update mechanisms was too slow to keep up with the 24 fps delivery rate required by the video benchmark. This is despite the fact that neither the client nor server was heavily loaded when running the video benchmark using ICA or RDP.

The second display update policy issue is whether a server-push or client-pull model drives the display update policy. In the server-push model, the server determines when to send a screen update to the client. In the client-pull model, the client sends a request to the server when it wants a screen update. A benefit of the client-pull model is that it provides a simple mechanism for adapting to the client processing speed and network speed available. Of the systems we considered, only VNC uses the client-pull model while all the other platforms use the server-push model.

Our measurements suggest that a server-push display update model is better at tolerating WAN latencies than a client-pull model. The problems with the client-pull model are illustrated by the performance of VNC on the video benchmark over Internet2. Over Internet2, the server must wait until the last display update is sent to the client and the client responds back requesting the next display update, which imposes a 66 ms RTT penalty. Even if the client were infinitely fast, the client-pull model would not allow the video to be delivered at 24 fps. VNC's client-pull model is the primary reason why its video benchmark performance is twice as bad over Internet2 versus a LAN, as shown in Figure 12. In contrast, Sun Ray avoids these problems by using an eager server-push display update model to send display updates immediately as video frames are rendered on the server for the video benchmark, resulting in the best video performance over Internet2. As multimedia applications become increasingly common and network bandwidth becomes increasingly available, we expect that the benefits of higher fidelity performance with a eager server-push display update policy will increasingly outweigh the benefits of bandwidth savings from lazy or client-pull display update models.

4. RELATED WORK

In addition to the six popular thin-client systems discussed in this paper, many systems for remote display have been developed. These include extensions to the systems considered such as low-bandwidth X (LBX) [4] and Kaplinsk's recent VNC tight encoding [15], as well as remote PC solutions such as Laplink [18] and PC Anywhere [24]. Because of space constraints and previous work [22, 36] showing that LBX, Laplink, and PC Anywhere perform worse than Microsoft Terminal Services for single-user workloads, we did not examine these systems and extensions as part of this study. While thin-client systems have primarily been employed in LAN workgroup environments, a growing number of ASPs are employing thin-client technology to attempt to host desktop computing sessions that are remotely delivered over WAN environments. Examples include services from FutureLink [5], Runaware [26], and Expertcity [12].

Several studies have examined the performance of a single thin-client system, in some cases in comparison to the X protocol. Danskin conducted an early study of the X protocol [10] by gathering traces of X requests. Wong and Seltzer have studied the performance of Windows NT Terminal Server, focusing on office productivity tools and web browsing performance [35,36]. Tolly Research has conducted similar studies for Citrix MetaFrame [34]. Schmidt, Lam, and Northcutt examined the performance of the Sun Ray platform in comparison to the X protocol [28] and reported results for Sun Ray focusing on office productivity application performance at various network bandwidths. None of these studies consider performance issues in WAN environments, nor do they compare across the range of thin-client platforms discussed here.

Few studies have been done that compare the performance of several thin-client systems. Yang, Nieh, et al. [22, 37, 39] examined the performance of several thin-client systems at various network bandwidths. This work does not consider the impact of network latency in WAN environments on thin-client systems. Our work addresses latency measurement issues not addressed in previous work–examining the broad space of underlying design choices that impact system performance. Previous work has also focused on the bandwidth efficiency of these systems. Our results show that efficient display encodings and compression algorithms are just one component of thin-client system performance.

While technology has changed, the vision of customers simply being able to rent their computing services from a public computer utility harkens back to the days of Multics [8]. Unlike Multics, ASPs are faced with supporting applications that are not just simple text programs but increasingly graphics and multimedia-oriented. However, further research needs to be done to enable computer utilities to effectively support multimedia applications in wide-area environments.

5. CONCLUSIONS AND FUTURE WORK

We have performed the first quantitative measurements to examine the impact of WAN latency on thin-client computing performance. We addressed the difficult problem of measuring proprietary, closed-source thin-client systems by using slow-motion benchmarking, which combines network monitoring with slow-motion versions of application benchmarks to provide accurate measurements of thin client performance. While our results demonstrate the feasibility of using thin-client computing for delivering computing services in a WAN environment, they also reveal that many of the design tradeoffs used in existing thin-client systems are inappropriate for such network environments. Our results demonstrate the importance of focusing on optimizing for network latency as opposed to bandwidth issues in designing thin clients. In this context, we show that minimizing the need for synchronized local client window system state, simpler, pixel-based display primitives, synchronized server-push display updates, and low-level forms of compression are surprisingly effective design choices. We examined these issues across a broad range of platforms and provide the first comparative analysis of the performance of these systems. These quantitative measurements provide a basis for future research in developing more effective thin-client systems to deliver wide-area computing services.

6. ACKNOWLEDGMENTS

We thank Monica Lam and Constantine Sapuntzakis for providing hardware resources at Stanford to enable us to conduct these experiments. We also thank Jae Yang for helping to install and configure the software used in our experimental testbed. This work was supported in part by NSF grant EIA-0071954 and an NSF CAREER Award. Internet2 characterization data obtained from the National Laboratory for Applied Network Research was sponsored by NSF Cooperative Agreement No. ANI-9807479.

7. REFERENCES

[1] Abilene Weather Map. http://hydra.uits.iu.edu/ abilene/traffic/.

[2] Active data query. http://amp.nlanr.net/Active/raw_data/cgi-bin/data_form.cgi.

[3] Active measurements. http://amp.nlanr.net/.

[4] Broadway / X Web FAQ. http://www.broadwayinfo.com/bwfaq.htm.

[5] Charon systems. http://www.charon.com.

[6] Citrix MetaFrame 1.8 Backgrounder. Citrix White Paper, Citrix Systems, June 1998.

[7] The Cloud. http://www.shunra.com.

[8] F. J. Corbato and V. A. Vyssotsky. Introduction and Overview of the Multics System. In *Proceedings of the Fall Joint Computer Conference*, volume 27, pages 185–196, June 1965.

[9] B. C. Cumberland, G. Carius, and A. Muir. *Microsoft Windows NT Server 4.0, Terminal Server Edition: Technical Reference*. Microsoft Press, Redmond, WA, Aug. 1999.

[10] J. Danskin and P. Hanrahan. Profiling the X Protocol. In *Proceedings of the SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Nashville, TN, May 1994.

[11] Description of Windows 2000 TCP Features. http://support.microsoft.com/support/kb/articles/Q224/8/29.ASP.

[12] DesktopStreaming Technology and Security. Expertcity White Paper, 2000.

[13] Etherpeek 4. http://www.wildpackets.com.

[14] i-Bench version 1.5. http://etestinglabs.com/benchmarks/i-bench/i-bench.asp.

[15] C. Kaplinsk. Tight Encoding. http://www.tightvnc.com/compare.html.

[16] D. Lake. Time to Learn Your ASPs. *The Industry Standard*, Nov. 2000.

[17] T. V. Lakshman, U. Madhow, and B. Suter. TCP/IP Performance with Random Loss and Bidirectional Congestion. In *IEEE/ACM Transactions on Networking*, volume 8, pages 541–555, Oct. 2000.

[18] *LapLink 2000 User's Guide*. Bothell, WA, 1999.

[19] J. Mahdavi. Enabling High Performance Data Transfers on Hosts. http://www.psc.edu/networking/perf_tune.html.

[20] T. W. Mathers and S. P. Genoway. *Windows NT Thin Client Solutions: Implementing Terminal Server and Citrix MetaFrame*. Macmillan Technical Publishing, Indianapolis, IN, Nov. 1998.

[21] Microsoft Windows NT Server 4.0, Terminal Server Edition: An Architectural Overview. Technical White Paper, 1998.

[22] J. Nieh and S. J. Yang. Measuring the Multimedia Performance of Server-Based Computing. In *Proceedings of the 10th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 55–64, Chapel Hill, NC, June 2000.

[23] J. Nielsen. *Multimedia and Hypertext: The Internet and Beyond*. Morgan Kaufmann Publishers, San Francisco, CA, Jan. 1995.

[24] PC Anywhere. http://www.symantec.com/pcanywhere.

[25] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1), Jan./Feb. 1998.

[26] Runaware.com. http://www.runaware.com.

[27] R. W. Scheifler and J. Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79–106, Apr. 1986.

[28] B. K. Schmidt, M. S. Lam, and J. D. Northcutt. The Interactive Performance of SLIM: A Stateless, Thin-Client Architecture. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP)*, volume 34, pages 32–47, Kiawah Island Resort, SC, Dec. 1999.

[29] A. Shaw, K. R. Burgess, J. M. Pullan, and P. C. Cartwright. Method of Displaying an Application on a Variety of Client Devices in a Client/Server Network. US Patent US6104392, Aug. 2000.

[30] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading, MA, 2nd edition, 1992.

[31] Sun Ray 1 Enterprise Appliance. http://www.sun.com/products/sunray1.

[32] Tarantella Web-Enabling Software: The Adaptive Internet Protocol. SCO Technical White Paper, Dec. 1998.

[33] A. Tirumala and J. Ferguson. Iperf. http://dast.nlanr.net/Projects/Iperf/.

[34] Thin-Client Networking: Bandwidth Consumption Using Citrix ICA. *IT clarity*, Feb. 2000.

[35] A. Y. Wong and M. Seltzer. Evaluating Windows NT Terminal Server Performance. In *Proceedings of the 3rd USENIX Windows NT Symposium*, pages 145–154, Seattle, WA, July 1999.

[36] A. Y. Wong and M. Seltzer. Operating System Support for Multi-User, Remote, Graphical Interaction. In *Proceedings of the USENIX 2000 Annual Technical Conference*, pages 183–196, San Diego, CA, June 2000.

[37] S. J. Yang and J. Nieh. Thin Is In. *PC Magazine*, 19(13):68, July 2000.

[38] S. J. Yang, J. Nieh, and N. Novik. Measuring Thin-Client Performance Using Slow-Motion Benchmarking. In *Proceedings of the 2001 USENIX Annual Technical Conference*, pages 35–49, Boston, MA, June 2001.

[39] S. J. Yang, J. Nieh, M. Selsky, and N. Tiwari. The Performance of Remote Display Mechanisms for Thin-Client Computing. In *Proceedings of the 2002 USENIX Annual Technical Conference*, Monterey, CA, USA, June 2002.