A Comparison of Thin-Client Computing Architectures

Technical Report CUCS-022-00
November 2000

Jason Nieh, S. Jae Yang, Naomi Novik
Network Computing Laboratory, Columbia University
{nieh, sy180, nn80}@cs.columbia.edu

Abstract

Thin-client computing offers the promise of easier-to-maintain computational services with reduced total cost of ownership. The recent and growing popularity of thin-client systems makes it important to develop techniques for analyzing and comparing their performance, to assess the general feasibility of the thin-client computing model, and to determine the factors that govern the performance of these architectures.

To assess the viability of the thin-client computing model, we measured the performance of five popular thin-client platforms running over a wide range of network access bandwidths. Our results show that current thin-client solutions generally work well in a LAN environment, but their performance degrades significantly when they are used in today's broadband environments.

We also find that the efficiency of the thin-client protocols varies widely. In some cases, the efficiency of the thin client protocol for web applications is within a factor of two of standard web protocols, while others are 30 times more inefficient. We analyze the differences in the various approaches and explain the impact of the underlying remote display protocols on overall performance.

1 Introduction

In the last two decades, the centralized computing model of mainframe computing has shifted to the more distributed model of desktop computing. But as these personal desktop computers become ubiquitous in today's large corporate and academic organizations, the total cost of owning and maintaining them can become unmanageable. In response to this threat, there is a growing movement to return to a more centralized and easier-to-manage computing strategy. The thin-client computing model is the embodiment of that movement.

The goal of the thin-client model is to centralize computing resources, with all the attendant benefits of easier maintenance and cheaper upgrades, while maintaining the same quality of service for the end user that could be provided by a dedicated workstation. In a thin-client computing environment, end users move from full-featured computers to thin clients, lightweight machines primarily used for display and input and which require less maintenance and fewer upgrades. Organizations then provide computing services to their end users' thin clients from high-powered servers over a network connection. Server resources can be shared across many users, resulting in more effective utilization of computing hardware.

While thin-client computing resembles a return to the days of mainframe computing, an important difference is that today's users can no longer be satisfied by dumb terminals that only input and output ASCII text. Users today are accustomed to more sophisticated graphical user interfaces which enhance their productivity. Thin clients must be able to support these graphical computing environments effectively to meet the demands of today's users.

Because of the potential cost benefits of thin-client computing, a wide range of thin-client platforms have been developed, including Citrix Metaframe, Microsoft Terminal Services, AT&T VNC, and others. [1, 4, 9, 12, 17, 19, 23] Some are designed specifically for use over high-bandwidth local area networks, while others attempt to provide quality service over slow network connections. Some application server providers (ASPs) are even offering thin-client service over wide area networks such as the Internet [6, 8, 16]. The growing popularity of these systems makes it important to develop techniques for analyzing their performance, to assess the general feasibility of the thin-client computing model, and to compare different thin-client platforms and determine the factors that govern their performance.

The main idea common to these modern thin-client platforms is the use of a remote display protocol. This key technology enables graphical displays to be served across a network to a client device, while all

application logic is executed on the server. Clients can be much simpler than in older systems such as X [18], since the window system executes on the server as well. Using such a protocol, the client transmits user input to the server, and the server returns screen updates to the client. In general, no unrecoverable state is stored on the client at all.

While many thin-client platforms and protocols have been developed, most of these systems and their protocols are proprietary, and few of the vendors have provided detailed performance measurements. It is difficult both to compare the performance of these solutions and to determine whether any of them can support realistic workloads. Of particular concern is the support they provide for web-based and multimedia-oriented applications, which have different and significantly higher resource demands than the office productivity applications for which many of these platforms were originally designed.

To assess the viability of the thin-client computing model, we measured the performance of a wide range of thin-client computing solutions in a variety of network operating conditions. We considered five different thin client platforms on four different operating systems running over a wide range of network access bandwidths. The thin client platforms we used to perform these head-to-head comparisons were Citrix Metaframe, Microsoft Windows Terminal Server, LapLink 2000, AT&T VNC, and Sun Microsystems Sun Ray. In particular, we focused on evaluating these thin client platforms with respect to their performance on popular Web and multimedia applications.

Our results show that current thin-client solutions generally work well in a LAN environment, their performance almost indistinguishable from that of full-featured desktop PCs, but they are inadequate for use over bandwidths currently available in wide area networks.

We also find that the efficiency of the thin-client protocols varies widely. In some cases, the efficiency of the thin client protocol for web applications is within a factor of two of standard web protocols, while others are 30 times more inefficient. We analyze the differences in the various approaches and explain the impact of the underlying remote display protocols on overall performance.

This paper is organized as follows: Section 2 presents the thin-client computing model in further detail and discusses the thin client platforms evaluated in our study. Section 3 details the experimental testbed and methodology we used for our study. Section 4 discusses our measurements and performance results. Section 5 describes related work. Finally, we present some concluding remarks and directions for future work.

2 Thin-Client Computing

The typical thin-client platform consists of a client application that executes on a user's local desktop machine and a server application that executes on a remote system. The end user’s machine can be a hardware device designed specifically to run the client application or simply a low-end personal computer. The remote server machine typically runs a standard server operating system, and the client and server communicate across a network connection between the desktop and server. The client sends input data across the network to the server, and the server returns display updates.

There are many design choices to be made within this basic framework, and as the results of our testing demonstrate, these can have a dramatic impact on the performance of these systems. We studied a wide range of current systems to reach our conclusions, including Citrix Metaframe and Microsoft Windows Terminal Services (both the Windows NT and Windows 2000 versions of these systems), AT&T VNC, Sun Ray, and LapLink. Table 1 summarizes the design of the systems we examined.

Perhaps most critical of these design decisions is the choice of encoding for display updates. It is important to note that simply sending raw pixel data directly to a display would be prohibitively expensive at current network bandwidths: even a 640x480 display with 8-bit pixel values would require over 100 Mbps at typical monitor screen refresh rates. Some thin-client platforms, such as Sun Ray and VNC, process updates to the display on the server and transmit only compressed pixel data representing the new display to the client. Others, like Citrix Metaframe and Microsoft’s Terminal Services, opt for a higher-level encoding that is more closely tied to the operating system’s windowing and display commands. In this case, graphics commands are transmitted from the server to the client, which is responsible for processing the updates.

Of similar importance is the timing of display updates. Most systems rely on the server to push out updates as they are generated, but some such as Metaframe and Terminal Services perform additional optimizations, bundling graphics commands together and discarding unnecessary interim commands before sending out an update. On the other hand, a client-pull technique is used in VNC, where the server refrains

Platform (Server OS)	Display Encoding	Screen Updates	Client Caching	Client Cache Size	Transport Protocol	Client Display
Microsoft Terminal Services (WinNT, Win2K)	Compressed graphics	Server buffer, pushed adaptively	Glyphs, small bitmaps in memory; large bitmaps on disk	1.5 MB RAM, 10MB disk in Win2K	TCP/IP	Up to 256 colors
Citrix Metaframe (WinNT, Win2K)	Compressed graphics	Server buffer, pushed adaptively	Glyphs, small bitmaps in memory; large bitmaps on disk	3MB RAM, Percent of disk (1% default)	TCP/IP	Up to 24-bit color
LapLink 2000 (WinNT)	DDI graphics, sent to client GDI	Each DDI call sent separately	Glyphs, small bitmaps	1024 objects	TCP/IP	256 colors
AT&T VNC (Linux)	2D run-length encoded pixels	Screen differences sent on client pull	None	N/A	TCP/IP	Up to 32-bit color
Sun Ray (Solaris)	Compressed pixels	Update pushed on each window system command.	None	N/A	UDP/IP	Up to 24-bit color

Table 1: Thin-client platform characteristics

from sending an update until a request is received from the client, and then sends only the most recent version of the display.

The client may also speed display updates by caching some amount of screen state locally, so it can perform at least some screen update operations without input from the server. Some clients may save the current contents of the display framebuffer so they only need incremental updates and can possibly re-use regions of the display that have merely been moved (as when a window is dragged across the screen). Others may cache high-level graphics primitives that are likely to be reused.

The quality of the display can also vary greatly among thin-client systems. Many systems have restrictions on resolution and color depth that limit the amount of data that must be transmitted in display updates but also leave the end user with a worse visual experience. Others such as the Sun Ray system can provide true color and high resolution graphics at the cost of increased bandwidth consumption.

The client machine can range from a full computer running an independent operating system to a simple network appliance like Sun Ray. Many of the other design decisions stem from this choice, which also has implications for the cost, reliability, and maintainability of the system. Simpler appliances have obvious advantages in cost, both in short-term purchase price and the long-term cost of upgrades and support, but are less tolerant of network failures and may make the system as a whole less scalable as a result of increased demands on the server. While we primarily focused on the basic performance of the platforms, this characteristic of any system is crucial when making an overall assessment.

3 Experimental Design

The goal of our research was to compare thin-client systems to assess their basic display performance and their feasibility in various network environments. We considered the five thin-client platforms discussed in Section 2: Citrix Metaframe, Windows Terminal Services, AT&T VNC, LapLink 2000, and Sun Ray. These platforms were chosen both because of their popularity as well as their architectural design differences. To evaluate their performance, we designed an experimental testbed and various experiments to exercise each of the platforms on single-user web-based and multimedia-oriented workloads at varying bandwidths.

In this section, we describe our hardware and software testbed setup in detail. We also describe the metrics we used to judge performance as well as the specific benchmarks and different computing environments used in our tests. We used several standard Web-based industry benchmarks for our application workloads, and we considered both latency in display updates and the amount of data successfully transferred to the client in assessing performance.

3.1 Experimental Testbed

Our testbed was designed to allow us to perform well-controlled experiments on the performance of the various thin-client platforms at varying bandwidths. Because of the proprietary nature of most of the platforms, the testbed was designed to enable us to employ non-intrusive measurement techniques for evaluating thin-client performance. 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 only runs on Sun machines.

Figure 1 shows our basic testbed configuration, which consisted of seven machines, five of which were active for any given test. The testbed consisted of two pairs of thin client/server systems, a network simulator machine, a packet monitor machine, and a benchmark server. The features of each system are summarized in Table 2, along with the SPEC95 performance numbers for the CPU in each system.

Four of the machines in the testbed were two sets of client/server systems, a Sun Ray thin client and thin server, and a PC thin client and thin server for running all of the other thin-client systems. The Sun thin server was used only for Sun Ray testing while the PC thin server was configured as a quad-boot machine to support the various PC thin-client systems and operating systems used for our experiments. As summarized in the hardware description in Table 2, there were some performance differences between the Sun and PC client/server systems. Overall, the Sun Ray client/server hardware was less powerful than the PC client/server hardware, particularly on the client-side in which the Sun Ray client used a lowly 100 MHz uSPARC CPU while the PC client uses a 300 MHz Pentium II CPU. As we discuss in Section 4, the large difference in client processing power was not a factor in our evaluations. However, a number of the experiments were limited by processing power on the server so the difference in the server processing power between Sun and PC server needs to be factored into those results.

As shown in Figure 1, the network simulator machine was placed in the middle of the network between the thin client and thin server machines to control the available network bandwidth between the thin client and server. This was done using a dedicated PC installed with two 100 Mbps Ethernet interfaces running The Cloud [3], a software network bandwidth simulator that could vary the effective network bandwidth between the two network interfaces. The thin clients and thin servers were separated from one another on isolated 100 Mbps networks which were then connected via the network simulator by connecting the server-side network to one of the network interfaces in the network simulator PC and connecting the client-side network to the other network interface. We refer to the server-side interface as the East gateway and the client-side interface as the West gateway. The Cloud software could then be used to vary the bandwidths on either of the two gateways from the maximum 100 Mbps to as little as 2400 bps. To ensure that this simulator did not itself introduce extra delay into our tests, we measured round-trip ping times with and without the simulator between the client and the server. There were no significant differences and round-trip ping times were roughly 0.6 ms in both cases.

A packet monitor was used in the testbed to monitor and record traffic on the network between the client and the server. The packet monitor used was a PC running Etherpeek 4 [7], a software packet monitor that timestamps and records all packet traffic visible by the PC. As shown in Figure 1, the packet monitor was generally attached to the client-side network to monitor client-side network traffic, though it could be moved to other parts of the testbed to monitor server-side traffic as well. In order to use the packet monitor to capture all packet traffic being sent in both directions between the thin client and server, hubs were used in the network in our testbed. Since traffic going through a hub is broadcast to all other machines

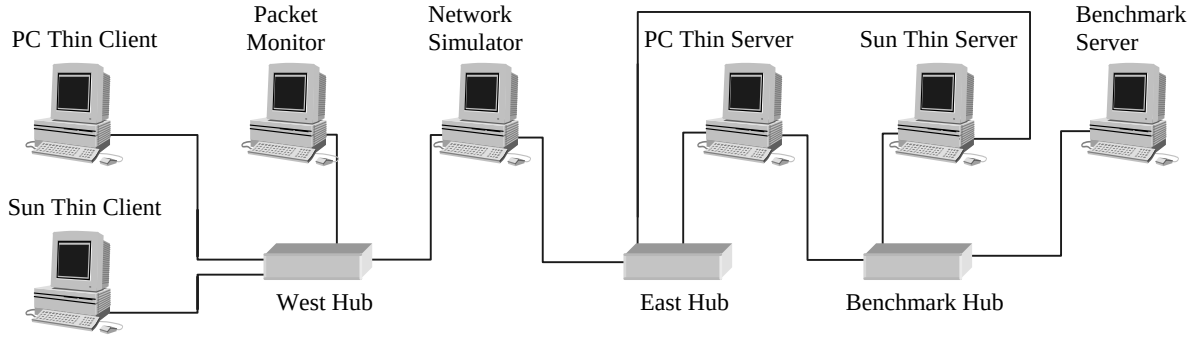


Figure 1: Testbed configuration

Role / Model	Hardware	OS / Window System	Software	CPU SPEC95
PC Thin Client NEC Direction SPL300	300 MHz Intel PII 128 MB SDRAM 6 GB Disk 3COM 3C905 100BaseT NIC Matrox G400 Max w/32 MB SDRAM	MS Win NT 4.0 Workstation SP6	Citrix ICA Win32 Client MS RDP4 Client MS RDP5 Client LapLink 2000 Client VNC Win32 3.3.3r7 Client Netscape Communicator 4.7	11.9 int 8.82 fp
Sun Thin Client Sun Ray I	100 MHz Sun uSPARC IIep 8 MB RAM 10/100BaseT NIC	Sun Ray OS	N/A	1.6 int 2.0 fp
Packet Monitor PC	400 MHz Intel PII 128 MB SDRAM 21 GB Disk 3COM 3C905 100BaseT NIC	MS Win NT 4.0 Workstation SP6	AG Group's Etherpeek 4	15.85 int 12.15 fp
Network Simulator NEC Direction SPL300	300 MHz Intel PII 128 MB SDRAM 6 GB Disk 2 3COM 3C905 100BaseT NICs	MS Win NT 4.0 Server SP6	Shunra Software The Cloud 1.1	11.9 int 8.82 fp
PC Thin Server Dell Dimension T550	550 MHz Intel PIII 128 MB SDRAM 21 GB Disk 2 3COM 3C905 100BaseT NICs	MS Win 2000 Advanced Server MS Win NT 4.0 Terminal Server MS Win NT 4.0 Workstation SP6 Caldera OpenLinux 2.4, Xfree86 3.3.6, KDE 1.1.2	Citrix Metaframe 1.8 Win NT Citrix Metaframe 1.8 Win 2000 MS Win NT 4.0 Terminal Server MS Win 2000 Terminal Services LapLink 2000 AT&T VNC 3.3.3r2 for Linux Netscape Communicator 4.7	22.2 int 15.0 fp
Sun Thin Server Sun Ultra-10 Creator 3D	333 MHz UltraSPARC IIi 384 MB RAM 9 GB Disk 2 10/100BaseT NICs	Sun Solaris 7, OpenWindows 3.6.1, CDE 1.3.5	Sun Ray Server 1.2_10d Beta Netscape Communicator 4.7	14.2 int 16.9 fp
Benchmark Server Dell Dimension T550	550 MHz Intel PIII 128 MB SDRAM 21GB Disk 3COM 3C905 100BaseT NIC	MS Win NT 4.0 Server SP6	Ziff-Davis i-Bench 1.01 MS Internet Information Server	22.2 int 15.0 fp

Table 2: Testbed summary

connected to the hub, the packet monitor could then be used easily to record packet traffic passing through any hub between the client and server by simply connecting the packet monitor to the respective hub.

We used a dedicated PC as a packet monitor to minimize the effect of our measurements when running the thin-client systems. A limitation of this network setup is that the hubs are half-duplex, so that traffic cannot be sent through the hub from client to server and from server to client at the same time. As we discussion in Section 4, since most data in these thin-client platforms is traveling from the server to the client in any case, it is unlikely that the half-duplex network would introduce significant delays for our experiments. Other options are possible, each with its disadvantages. An alternative would be to run a packet monitor on the thin client or thin server, but Etherpeek is highly resource-intensive and would undoubtedly adversely affect our performance results. Furthermore in the case of the Sun Ray thin client device, it is not possible to run a packet monitor on the client. Another alternative would be to use port-mirroring switches to support full-duplex network connections, but mirroring typically would only allow monitoring of either client to server traffic or vice versa, not both at the same time.

Finally, we also had a separate benchmark server, which was used for serving web-based workloads to the thin client platforms. The benchmark server was used to run the Ziff-Davis i-Bench benchmark suite [10]. To ensure that network traffic from the benchmark server did not interfere with the network connection between thin client and thin server, the benchmark server was connected to the testbed using a separate hub, as shown in Figure 1. Each thin server had two 100 Mbps network interfaces, one connected to the network simulator going to the client and the other connected to the benchmark server.

3.2 Application Benchmarks

To measure the performance of the thin-client platforms, we used a simple Java latency benchmark and a selection of benchmarks from the Ziff-Davis i-Bench benchmark suite, version 1.01. The Java latency benchmark was used to measure the latency of basic operations on a thin-client platform, such as responding to a single keystroke. The i-Bench benchmarks used were the Web Text Page Load and Flash

benchmarks, which can be used to provide a measure of web-based and multimedia-oriented application performance. All of the benchmarks were designed to be executed from within a web browser to provide a common application environment across different platforms. Figure 2 shows screenshots of these three benchmarks in operation.

Network Type Simulated	Bandwidths Tested
LAN	100 Mbps 10 Mbps 4 Mbps
T1	1.5 Mbps
DSL	768 Kbps 512 Kbps
ISDN	256 Kbps 128 Kbps

Table 3: Bandwidths tested

The latency benchmark used was a small Java applet that permitted us to run four separate tests: typing a character, scrolling text, filling a screen region, and downloading an image. The typing test took a single keystroke as input and responded by displaying a 12-point capital letter 'A' in sans serif font. The scrolling test involved scrolling down a page containing 451 words in 44 lines in 12-point sans serif font, with 21 of the lines displayed in a 160x316 pixel area at any one time. The screen fill test would respond to a mouse click by filling a 216x200 pixel area with the color red. The image download test would respond to a mouse click by downloading and displaying a 37 KB image in GIF format at 320x240 pixels in size and at a resolution of 72 dpi.

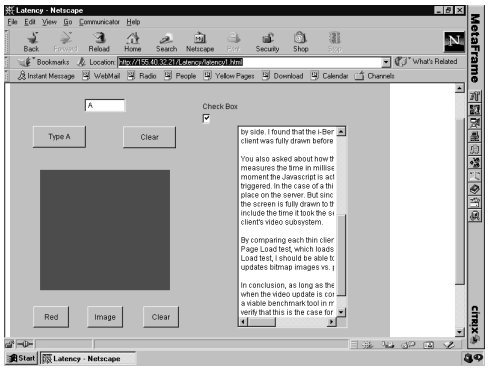
The Web Text Page Load benchmark is a Javascript-controlled load of a sequence of 54 web pages from the i-Bench benchmark server. Each page is downloaded and then programmatically scrolled down 200 pixels. The pages contain both text and bitmap graphics, with some pages containing more text while others contain more graphics. Some common elements were included on each page, including a blue left column, a white background, a PC Magazine logo and other small graphics. The Javascript cycles through the page loads twice and reports the elapsed time in milliseconds as a separate web page. Including the final results page, a total of 109 web pages are downloaded during this test.

The Flash benchmark streams a 98 KB Macromedia Flash animation clip from the i-Bench benchmark server. The animation clip uses vector graphics and contains 315 550x400 frames. The test measures the frame rate observed during the playback of the animation.

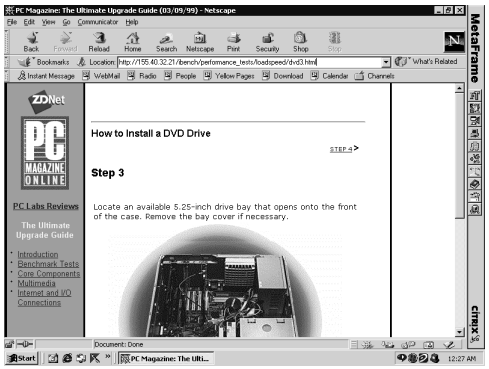
3.3 Measurement Methodology

For our evaluation study, we ran the three application benchmarks on each of the platforms and used the network simulator to vary the network bandwidth between client and server to examine the impact of network bandwidth on thin-client performance. All tests were run on each system at the bandwidths listed in Table 3. Eight different network bandwidths were considered, representing LAN, T1, DSL, and ISDN bandwidth connections. We focused our evaluation on network bandwidth and did not consider the different network latencies associated with different network technologies, which is beyond the scope of this paper.

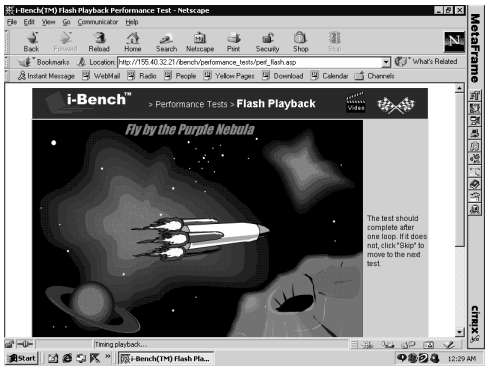
Two key issues that needed to be addressed in measuring the performance of thin-client platforms on different operating systems were application environment differences and the proprietary black-box nature of almost all the thin-client systems. To minimize application environment differences, we used common thin-client configuration options and common applications across all platforms whenever possible. For all of our experiments, the video resolution of the server was set to 1024x768 with 24-bit color and the thin client was set to 800x600 resolution with 8-bit color, except for Sun Ray. 8-bit color depth was used for the thin clients to provide a fair comparison with those clients that could not support 24-bit color. Windows Terminal Services and LapLink only support up to 8-bit color, and Citrix Metaframe only introduced 24-bit color support a few weeks ago. For Sun Ray, the client color depth was set to 24-bit color because the Sun



Java Latency Test



Web Page Load Test



Flash Video Test

Figure 2: benchmark screenshots

Ray display protocol is based on a 24-bit color encoding. Displaying in 8-bit color requires the Sun Ray server to convert all pixels to 24-bit mode before they are sent over the network. As a result, displaying in 8-bit color reduces the display quality, increases the server overhead, but does not reduce the bandwidth requirements. Since disk caching was turned off by default in those platforms that supported it and disk caching was not supported on VNC and Sun Ray, we did not enable disk caching for our comparative measurements.

Since all of the benchmarks can be launched from a web browser, we used Netscape Navigator 4.7 to execute all of the benchmarks, 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 800x600 in size, so the region being updated was the same on each system. Nevertheless, Netscape on Windows (Windows NT, Windows 2000) performs somewhat differently from Netscape on Unix (Linux, Solaris). For instance, on Unix, fonts appear smaller by default and a gray blank page appears between page downloads which does not appear on Netscape on Windows. These effects would tend to increase the amount of data that would need to be transferred on screen updates using a Unix-based thin-client platform. Our experiences with various thin-client platforms indicate that these effects are minor in general, but should be accounted for when considering small thin-client performance differences across Unix and Windows systems.

Since the user-perceived performance when running on a thin client is based on what is displayed on the client-side of the system, we measure performance of the thin-client platforms tested at the client-side. However, this cannot be done by simply running a measurement application program on the thin-client system since all applications are executed on the server-side. Therefore, application performance results reported by the application benchmarks may not be an accurate reflection of client-side performance. For instance, the latency seen by an application running on a thin-client system is generally just the latency on the server-side and does not include the end-to-end latency that a user would perceive at the client side. Similarly, the frame rate reported by the Flash benchmark measures the frame rate as rendered on the server, but if not all the frames are sent to the client for display, then the frame rate reported by the Flash benchmark would give an overly optimistic view of performance. This problem is exacerbated by the fact that all of the thin-client platforms except for VNC are proprietary black box systems, making it harder to gather client-side performance measurements.

Since we could not peer into the black-box thin-client, our solution to this problem was to use the packet monitor to gather measurements on the client-side. To measure the results from the latency benchmark from user input to client output, we used 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 was used as a measure of latency. For measuring the Web and Flash benchmarks, we combined application performance measurements on the server with network traffic measurements gathered by the packet monitor on the client-side. The server-side application measurements indicate application performance. The client-side data transfer measurements at different network bandwidths can indicate how much of the screen updates generated on the server-side are actually delivered to the client. For instance, if the Flash benchmark reports the same frame rate at both 100 Mbps and 128 Kbps network access bandwidths but delivers 3 times more data at the 100 Mbps bandwidth, the real client-side performance at 100 Mbps is 3 times better than the performance at 128 Kbps. An underlying assumption here is that the thin-client platform does not adaptively compress screen updates based on network bandwidth. However, none of the platforms considered in this study perform such adaptive compression.

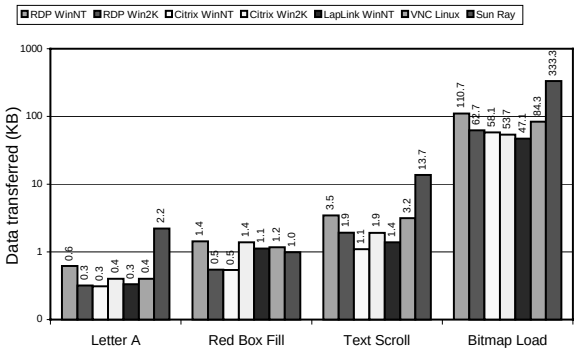
As a baseline measure, we also conducted the same set of experiments using the PC thin server directly as a normal PC “fat” client. We found that the baseline results for the benchmark tests were fairly similar for the PC running Windows NT, Windows 2000, and Linux. Linux performed slightly worse at lower bandwidths and slightly better at higher bandwidths, but in general all three were close enough that we discuss the results in the Windows NT case in Section 4.

4 Measurements and Results

We ran each of the three application benchmarks on the baseline PC platform and the five thin-client platforms on different operating systems. The eight specific platform configurations considered were PC running Windows NT 4.0 Server (WinNT Server), Microsoft Terminal Server RDP4.0 on Windows NT 4.0 Terminal Server Edition (RDP WinNT), Microsoft Terminal Services RDP 5.0 on Windows 2000 (RDP Win2K), Citrix Metaframe 1.8 on Windows NT 4.0 Terminal Server Edition (Citrix WinNT), Citrix

Metaframe 1.8 running on Windows 2000 (Citrix Win2K), LapLink 2000 on Windows NT 4.0 Terminal Server Edition (LapLink WinNT), AT&T VNC on Linux (VNC Linux), and Sun Ray. We also ran LapLink 2000 on Windows 2000 and VNC on Windows NT and Windows 2000. However, LapLink 2000 did not function at all on our Windows 2000 configuration and the Windows VNC server implementation is known to be implemented poorly [25] and performed much worse than the Linux version of VNC. As a result, we did not report detailed results for LapLink 2000 on Windows 2000 or VNC on Windows. Below we provide some basic characterization of the client-side component of each thin-client system, discuss the results from the latency benchmark testing, and describe the results from the Web and Flash benchmark tests.

Chart 3: Latency Test
Data Transferred



4.1 Memory Footprint

To provide an indication of how thin a thin client is, Chart 1 shows the sizes of the various clients in the thin-client platforms, both in terms of the size of the primary executable file and its memory footprint. Except for Sun Ray which is a hardware thin client, all of the other clients were measured running on the PC client running Windows NT 4.0 Workstation. The VNC client was the smallest by far with a memory footprint of less than 300 KB and an executable file of just 172 KB. The Sun Ray client had only 8 MB of memory and used just 2 MB during execution [19]. At the other end of the spectrum, the Citrix client required almost 10 MB of memory, which is comparable to the memory requirements of a Netscape web browser. LapLink also had a large file size and a large memory footprint, which was not surprising when considering that the same software is used for both the client and the server. In general, the resources required by clients in platforms that use graphics encoding for display updates were significantly higher than those needed by clients in the platforms that use pixel encodings (Sun Ray and VNC).

4.2 Latency Benchmark Results

We ran the latency benchmark using each thin-client platform at the full 100 Mbps network bandwidth to measure the latency of each of the thin-client platforms when network capacity was not the limiting factor. We measured both the latency and the data transferred for each of the four operations, draw letter, fill red box, scroll text, and load bitmap. These results are shown in Charts 2 and 3, respectively. Both charts show substantial variations in the latency and data transfer measurements across different platforms.

To achieve good subjective performance, the latency between user input and system response should be below the threshold of human perception. For simple tasks such as typing, cursor motion, or mouse

Chart 2: Latency Test
Time To Completion

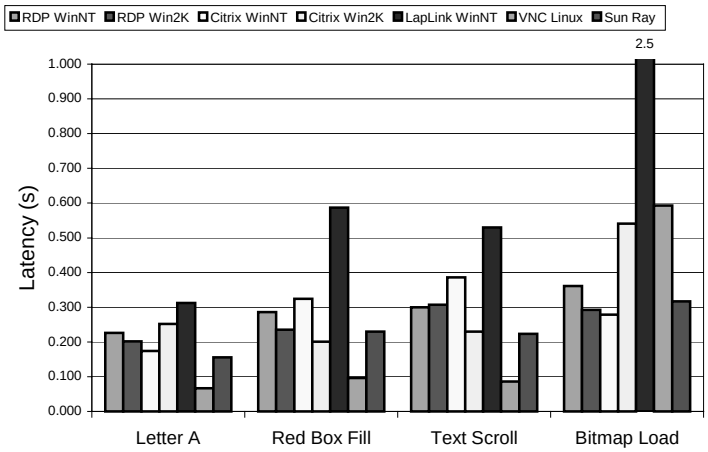
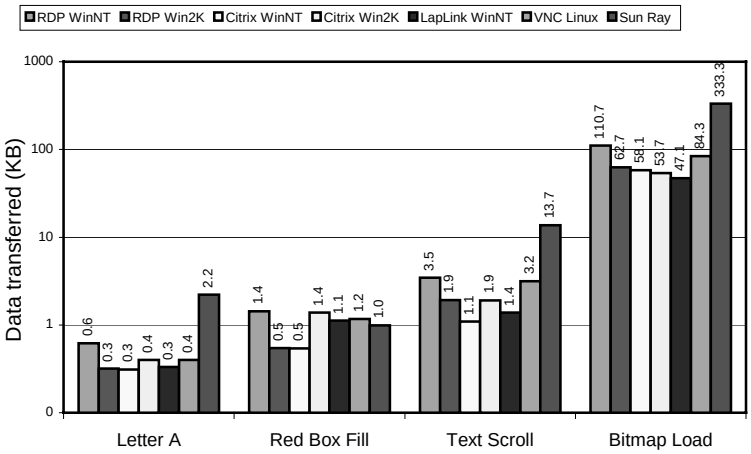


Chart 3: Latency Test
Data Transferred



selection, system response time should be less than 50-150 ms to keep users from noticing a delay [20]. Chart 2 shows most of the thin-client platforms performed all operations in less than a second, but not in less than 150 ms. Only VNC and Sun Ray were able to complete any operations within 150 ms. VNC completed the basic draw letter, fill box, and scroll text operations in less than 100 ms while Sun Ray completed the draw letter operation in roughly 150 ms. Note that both VNC and Sun Ray ran using Unix servers and employed a pixel-based encoding approach. Among the Windows-based thin clients, Citrix Win2K had the best performance overall except for loading the image bitmap, in which it was one of the worst. LapLink WinNT had by far the worst performance of all the systems, requiring more than 2.5 seconds just to display a small bitmap image. We examined the load on the LapLink server and found that the server was completely loaded when generating display updates, indicating that the server load and not the bandwidth was the limiting factor in this case.

Comparing Chart 2 with Chart 3, we can see that the latency and the data transferred were not at all correlated in many cases. While VNC and Sun Ray had less latency for most of the tests, they also required more data transfer. For instance, Sun Ray sent the second largest amount of data for the load bitmap test, but had one of the smallest latencies. Considering that Sun Ray is operating in 24-bit display mode, it is not surprising that it sends more data than the other platforms. On the other hand, LapLink WinNT sent the smallest amount of data for the load bitmap test, but had the largest latency. We also see that the load bitmap test resulted in the largest variation in data transfer across platforms. In loading the 37K bitmap, the amount of data transferred varied from 47K for LapLink WinNT to over 300K for RDP WinNT.

There are marked differences between the versions of RDP and Citrix running on different Windows operating systems. For RDP, there was a dramatic difference between the performance of RDP WinNT and RDP Win2K. The difference is due to improvements in the RDP protocol from RDP 4.0 used in Windows NT 4.0 Terminal Server Edition to RDP 5.0 used in Windows 2000, as well as problems in the implementation of the compression feature in RDP 4.0. For Citrix, there is a smaller performance difference between Citrix WinNT and Citrix Win2K. However, for all of the operations except the basic draw letter operation, the version of Citrix that required more data transfer on the operation also resulted in lower latency. For instance, in the scroll text test, Citrix WinNT took about 50 percent longer than Citrix Win2K to perform the operation, but sent almost half as much data as Citrix Win2K.

While latency and data transfer requirements were not correlated at this high network bandwidth, we would expect the data transfer requirement would play a more significant role at lower network bandwidths. Our results suggest that pixel-based encoding approaches may be less complex and hence faster, but may also be less efficient in terms of data transfer requirements. As a result, the graphics-based encoding approaches that require less data transfer are more likely to yield better performance at lower bandwidths.

4.3 Web Page Load Benchmark Results

We ran the Web benchmark on each of the thin-client platforms using network bandwidths from 128 Kbps to 100 Mbps, as listed in Table 3. We also ran the Web benchmark on the baseline PC platform for comparison. For each run, we measured the total time required to display all 109 web pages of the Web benchmark and logged complete packet traces using the packet monitor. We found that at high bandwidths, the page load times were limited by server speed as the server was completely busy. Since server speed was the limiting factor at high network bandwidths, we normalized the page load times across PC and Sun thin-client platforms according to the relative SPEC95int performance of the PC and Sun servers to provide a more fair comparison. Based on the SPEC95int numbers in Table 2, the page load times for the Sun server were normalized by a factor of 0.64. The total normalized web page download times and total data transferred for each run are shown in Charts 4 and 5, respectively.

As shown in Chart 4, almost all of the platforms completed the Web benchmark in less than 50 seconds at network bandwidths of 4 Mbps or greater, corresponding to an average of less than half a second per page. Though there were quantitative differences in performance among the platforms, we found that a download speed of less than 50 seconds provided a subjectively adequate web browsing experience.

The one exception was LapLink, which required almost an order of magnitude more time to complete the Web benchmark. However, as shown in Chart 5, the excess load time for LapLink was not due to excess data transfer requirements. LapLink required at most twice as much data transfer as the other Windows-based thin-client platforms, which is not surprising given that all of the Windows-based thin-client systems are based on some form of DDI graphics encoding. The key difference between LapLink and the other platforms is that it does not buffer any draw commands at the server, but instead sends each individual command to the client. The server then busy waits until the client informs the server that it has

Chart 4: Web Page Load Time

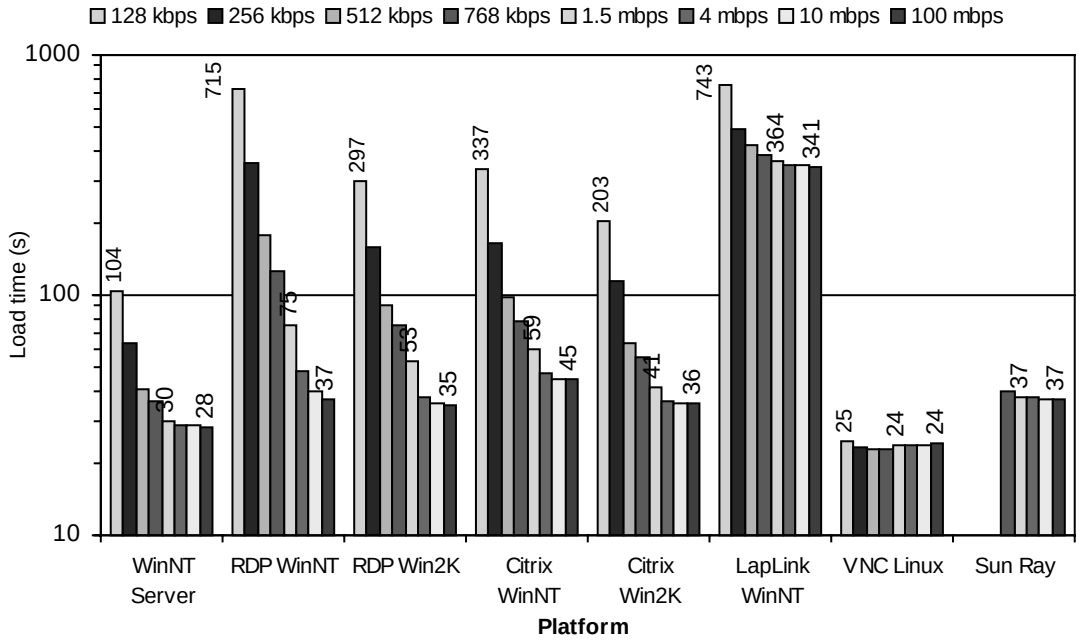
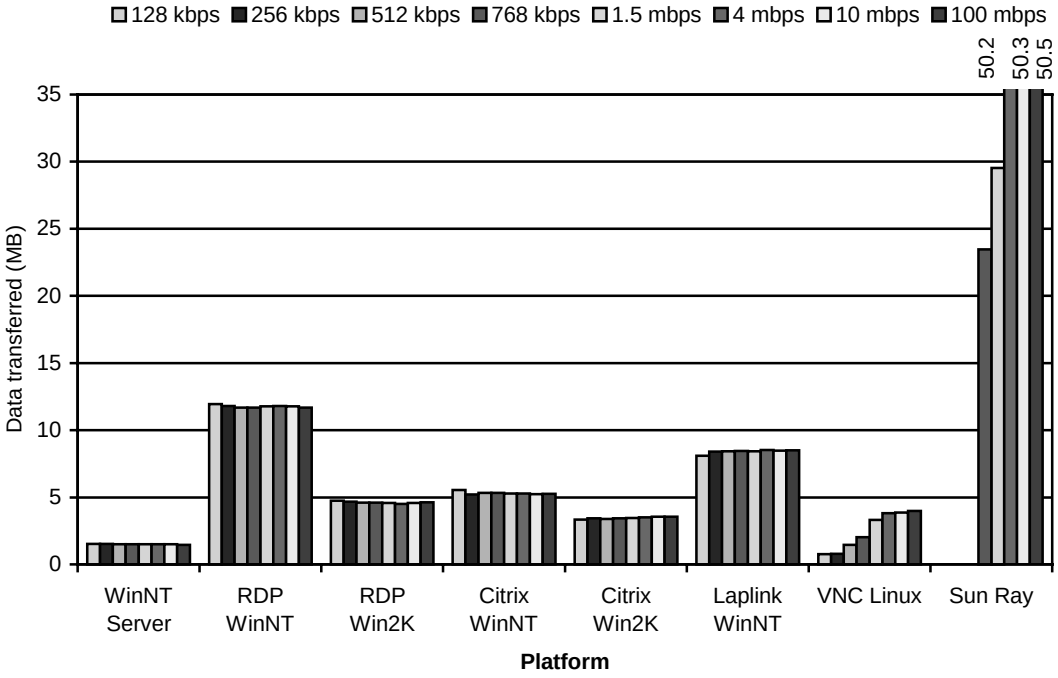


Chart 5: Web Page Data Transferred



completed the respective draw command. The much longer draw command latency causes LapLink performance to suffer.

At network bandwidths below 4 Mbps, the performance of the thin-client platforms begins to degrade. As shown in Chart 4 and 5, RDP and Citrix behave in a similar manner to the baseline PC client by taking longer to complete the Web benchmark as the network bandwidth decreases, but continuing to send the same amount of data even at lower bandwidths. On the other hand, VNC and Sun Ray do not take longer to complete the Web benchmark at lower bandwidths, but instead lose data resulting in missed or incomplete screen updates. As shown in Chart 5, VNC and Sun Ray transfer less and less data at the lower network bandwidths unlike the other platforms which all transfer roughly the same amount of data. As described in Section 2, the VNC server only sends display updates in response to client requests. When update requests

come slowly, as in a low-bandwidth environment, fewer updates are sent. As a result, the screen is refreshed very infrequently, with many intermediate screens skipped. Subjectively, the method of skipping screen updates in VNC provided a lower quality web browsing experience than sending all the screen updates and simply taking longer to load the web pages, as was done in RDP and Citrix. While Sun Ray does not use a client pull screen update method like VNC, it also skipped screen updates at lower bandwidths. The reason for this was that at lower bandwidths, the network becomes congested and the UDP packets used to send the screen updates end up being lost. Due to excessive packet loss, Sun Ray simply ceased to function properly at bandwidths below 768 Kbps, which is why there are no Sun Ray results shown in Charts 4 and 5 at bandwidths below 768 Kbps.

At the lower network bandwidths, the amount of data transfer required by each platform becomes the dominant performance factor. At the lower bandwidths, the baseline WinNT server performed the best of the platforms because it required the least amount of data transfer. Among the thin-client platforms, Citrix Win2K performed the best and was the only thin-client platform to complete the Web benchmark without losing data in less than 50 seconds at the 1.5 Mbps network bandwidth. At the other end, Sun Ray required the largest amount of data transfer to send it screen updates from server to client. The data transferred using Sun Ray was more than an order of magnitude more than using Citrix. Since Sun Ray operates using a 24-bit display protocol, it is not surprising that it requires more data transfer to send the screen updates. Nevertheless, even if we normalize the Sun Ray data transfer by a factor of 3 to account for the difference between its 24-bit mode and the 8-bit color used in the other platforms measured, Sun Ray would still send more than three times as much data as Citrix. While a comparison of Citrix to Sun Ray would suggest that the graphics-based encoding approach is more efficient for encoding screen updates, we note that VNC sends less data than RDP WinNT, RDP Win2K, and Citrix WinNT. Instead, the VNC results indicate that for Web-based applications, a pixel-based encoding approach can encode screen updates with comparable efficiency as graphics-based encoding approaches.

In comparing the data transfer requirements of the thin-client platforms to the WinNT Server baseline case, both Citrix Win2K and VNC required only about twice as much data to send their screen updates as compared to sending HTML and JPEG and GIF images in the WinNT Server baseline case. This is impressive, particularly considering that the standalone WinNT Server machine is largely receiving plain text and a few graphic images. As network technologies continue to improve, this factor of two difference in encoding efficiency may be less significant. Note though that the thin-client platform data transfer requirements scale with screen size while the WinNT Server baseline data transfer requirements scale with the total web page size. For larger screen displays, the thin-client platforms would be less efficient relative to the WinNT Server baseline.

When considering the data transferred, we found that the efficiency varied dramatically. As noted, Sun Ray sent almost 25 times the baseline amount of data, and RDP NT was 8 times less efficient. But at least some of the platforms were within a factor of 2 of the baseline performance. At higher bandwidths, the

Chart 6: Display Packet Size

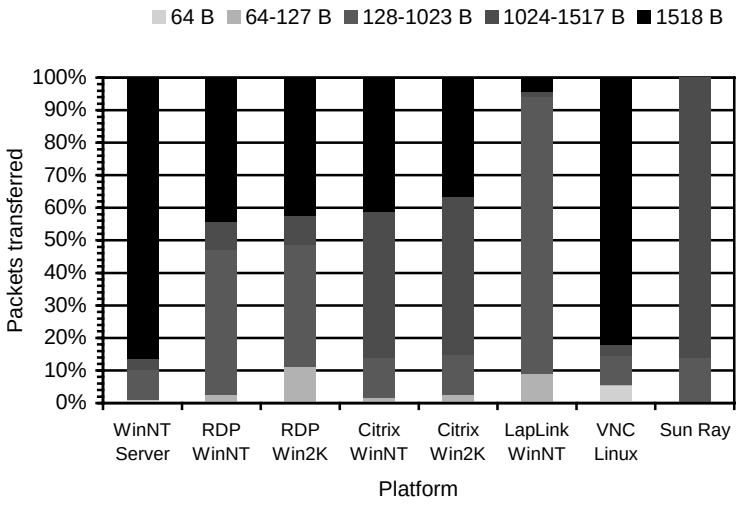
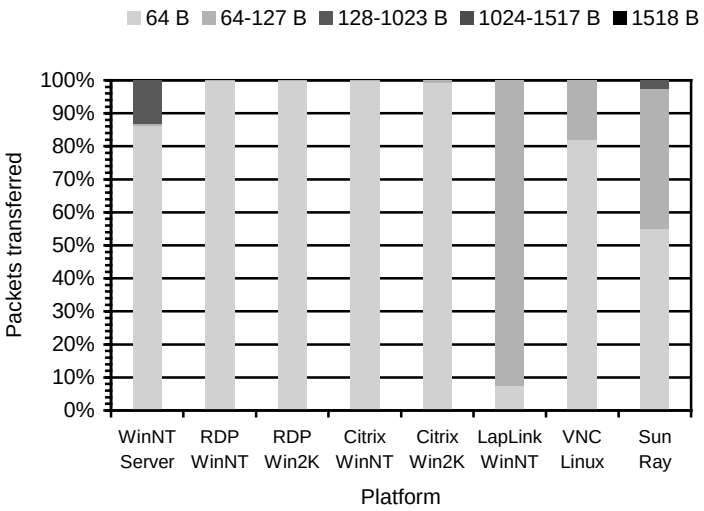


Chart 7: Control Packet Size



differences in efficiency did not greatly affect the download time. Though RDP NT was much less efficient than Citrix 2K, for instance, at 100Mbps their download times were almost identical. However, as bandwidth decreased, the more efficient platforms performed significantly better.

As no user input was entered during the course of the test, all of the data sent from the client represented control information. The amount of control data sent was smaller than the amount of display data, as we would expect, but some of the platforms transmitted surprisingly large amounts of control data. Control data is 10% of the data sent by LapLink, one possible source of that platform's inefficiency and long download times. Sun Ray, on the other hand, sends almost no control data even though it sends a great deal of display data, reflective of the simplicity of the client.

It is also instructive to consider the number and size of the packets being sent between the client and server. Charts 6 and 7 show the size of the packets in each direction, while Charts 8 and 9 show the relative amounts of display and control packets and data sent for each platform.

Chart 8: Display vs. Control Packets

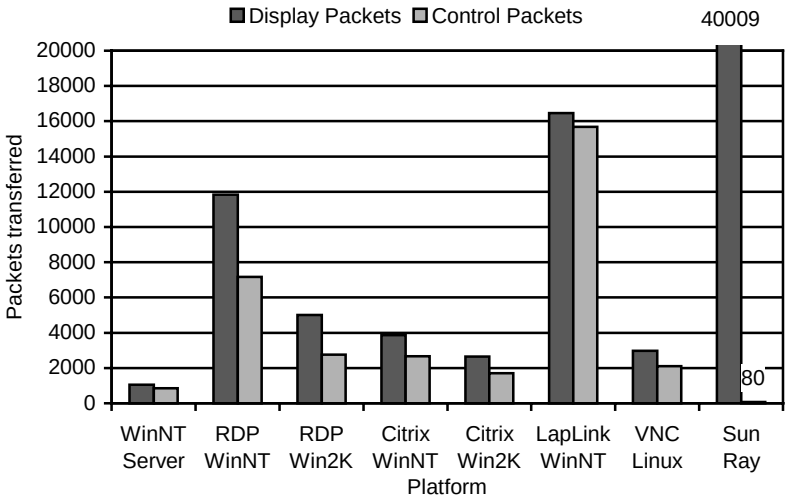
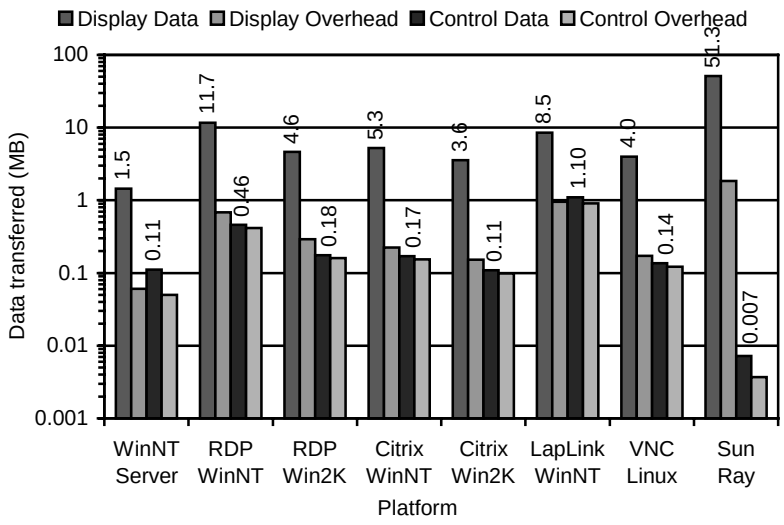


Chart 9: Display vs. Control Data



As no user input was entered during the course of the test, all of the data being sent from the client to the server represented control information. The amount was significantly smaller than the amount of display data, as we would expect, but some of the platforms send surprisingly large amounts of control data. Control data accounts for 10% of the data and 50% of the packets sent by LapLink, another possible source of that platform's inefficiency and long download times.

We found another explanation for LapLink's poor performance in the packet sizes it sends: most display data is sent in relatively small packets, while control data is sent in relatively large packets. The other platforms pack the large quantities of display data much more efficiently, and send many fewer packets to boot. This is clearly an important factor in achieving reasonable efficiency and performance.

4.4 Flash Benchmark Results

We ran the Flash animation test on each of the thin-client platforms using the network bandwidths listed in Table 3, and also ran the benchmark on the baseline platforms for comparison. Charts 10 and 11 display our results. As the packet size and ratio of display data to control data for these tests closely resembled the Web Page Load results, we have not included charts of these results.

We found that a frame rate close to 18 fps produced good subjective results, with smooth display, no skipped screens, and no tearing or jerky movement. The animation was essentially unusable at frame rates below 8 fps. The 98KB animation downloaded completely to the server prior to playback, so the bandwidth did not affect the frame rate on the baseline. However, we examined the time required for this download in

Chart 10: Flash Test Frame Rate

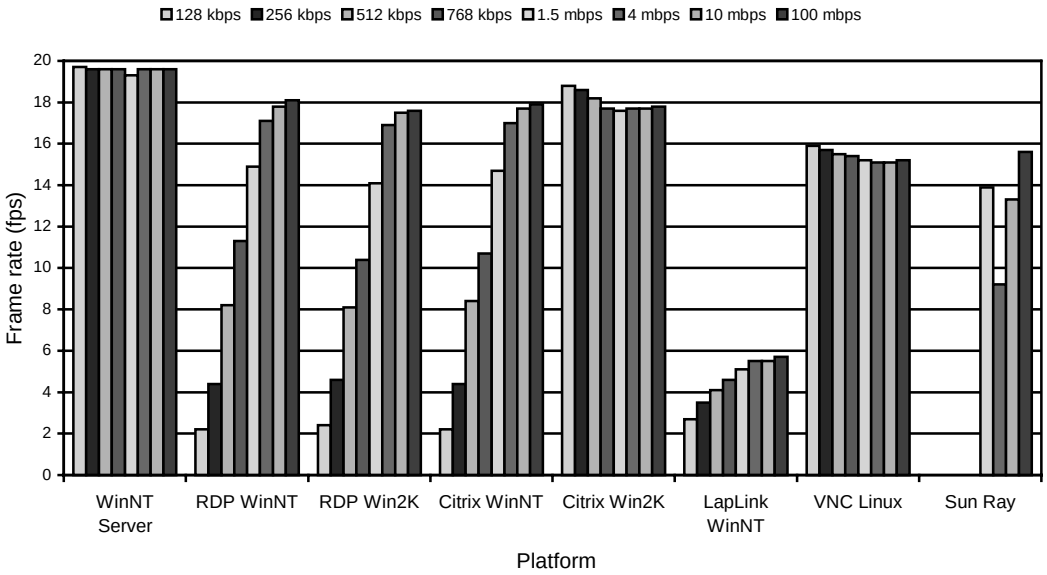
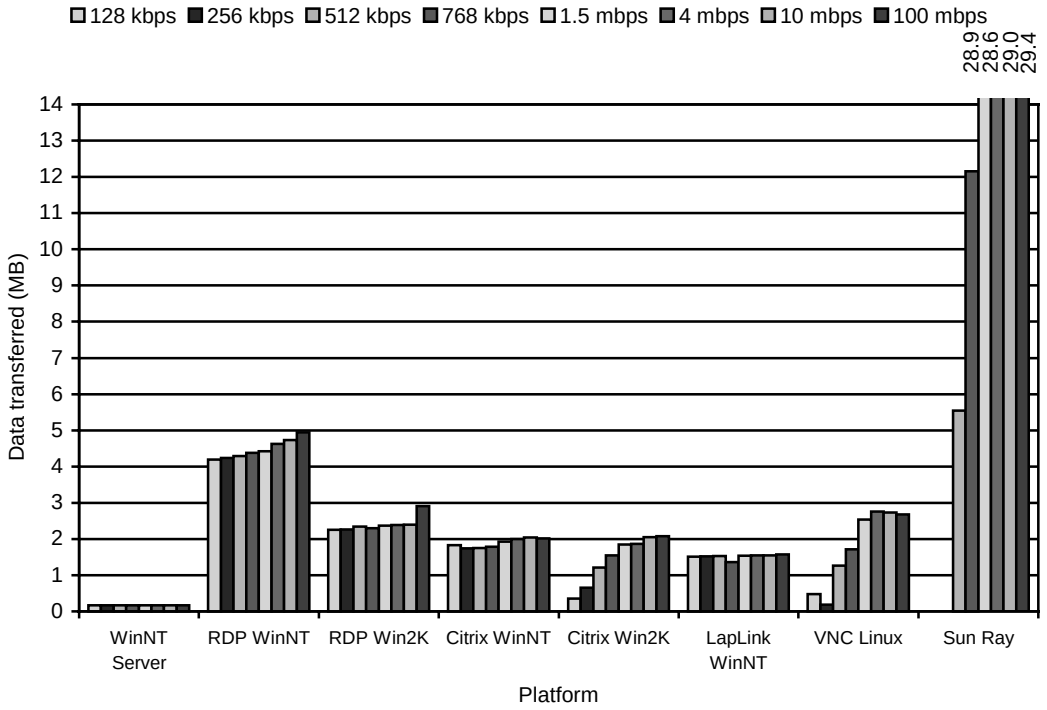


Chart 11: Flash Test Data Transferred



the baseline case and found that there were no significant delays except a brief one when the bandwidth was limited to 128 Kbps.

In general, our results supported the conclusions drawn from the Web Page Load test and the latency testing. All the platforms experienced a similar fall-off in performance at T1 bandwidths or below, confirming our findings that LAN bandwidths are required to support multimedia applications.

Terminal Server and Metaframe again showed good performance at the LAN bandwidth level, with slight tearing apparent at the lower end of that spectrum for both of them. At lower bandwidths, however, playback became jerky and slow. LapLink's performance remained very poor, both quantitatively and subjectively, regardless of the bandwidth. All three of these platforms, with the exception of Metaframe under Windows 2000, varied mainly in the speed of the playback; they only skipped a few frames if any and just played back slowly at lower bandwidths.

The SunRay system was subject to significant bandwidth constraints, sending almost 30MB of data, and only performed well at 100Mbps. Both the frame rate and data transferred degraded at any lower bandwidths. Our charts do not include frame rate results for broadband or lower bandwidths, as the benchmark did not even complete below the T1 level. The sheer quantity of data sent by the Sun Ray for such a small animation is surprising, and suggests that even at LAN levels, this platform could not support multiple users of video or animation applications since even nine users would completely fill the pipe. However, there are some undocumented low-bandwidth options in the Sun Ray platform that we did not use in testing, which might make the platform perform better at lower bandwidths.

As with the web test results, VNC was only slightly less efficient than the platforms with graphics-based encodings, so again the Sun Ray results are not sufficient cause to rule out a pixel-based encoding technique. VNC maintained an almost constant rate for the benchmark at all bandwidths, but dropped many screens on the client's display at the low end of the range. Its performance was only comparable in smoothness to the baseline when the data transferred reached a plateau. Again, this is due to the client-pull method VNC uses for updates.

Interestingly, Citrix Win2K had similar behavior, although Citrix WinNT did not. Citrix Win2K maintained a constant frame rate from the i-Bench server to the thin server but dropped many frames in delivering the animation to the client, just as VNC did. Its performance is better measured by examining the total amount of data transferred instead of the frame rate and was subjectively good at LAN bandwidths. The update policy is known to have changed for the new version of Citrix on Windows 2000, and this change indicates that part of the policy change has been to preserve the speed of updates at the expense of data loss in some cases.

While an adaptive update policy like this clearly can preserve speed, overall performance still suffers the same kind of fall-off at below-LAN bandwidths, as the data transfer results for Citrix Win2K and particularly VNC show. In addition, in video and animation applications like the Flash animation, the number of frames dropped may be a better measure of quality than the speed at which the application executes. Losing data in favor of preserving speed is a poor choice for an application like this as long as it is still usable. As we can see from the frame rate results, RDP and Citrix WinNT all had usable if not good frame rates even when bandwidth was as low as 512 Kbps. By that bandwidth level, Citrix Win2K and VNC had both dropped almost half of the data transmitted. The animation became unusable more quickly under the adaptive update policies in this situation.

Citrix Win2K did not drop data in the Web Page Load testing as VNC did, which suggests that Citrix protocol takes factors other than the available bandwidth into consideration when deciding whether or not to discard display updates. Future work will investigate what effect server load and other conditions have on the update behavior.

4.5 Summary

Overall, our results indicate that current thin-client platforms are viable alternatives for a typical LAN environment. Their requirements, even for many multimedia applications, are well within 10 Mbps, so almost all of the platforms examined would work well under LAN conditions.

However, it is clear that these platforms cannot support typical, multimedia-intensive workloads over wide area networks. Even the best protocols require a pipeline of 4 Mbps or better to prevent the network from becoming a bottleneck. Even at the T1 level their performance degrades significantly, and at lower broadband levels many of them become unusable. This has major implications for the application service provider (ASP) model, as many of today's ASPs are attempting to use these thin-client platforms to provide computing services over the Internet. Our results show that this is currently impractical.

This is not to say that it is impractical for all purposes. Our tests were deliberately graphics-and-multimedia intensive, and typical office productivity software may function well over WANs. But multimedia applications ran smoothly at even DSL bandwidths on our baseline tests, so excellent performance with these kinds of applications is possible over broadband connections using standard desktops. Current users are unlikely to be willing to sacrifice their multimedia environments and applications if another choice is available.

We found that the efficiency of the various protocols varies widely, with more than an order of magnitude separating them. Some protocols are highly efficient, within a factor of 2 of the stand-alone desktop performance, and this efficiency is reflected in their improved performance at lower bandwidths. We identified several factors that affect efficiency, including the ratio of control data to update data, and packet size. These are causes for the poor performance of LapLink, which sends a large amount of control data and packetizes data inefficiently.

Our results also showed that higher-level encodings are not necessarily ideal for graphics-intensive multimedia applications with many rapidly changing images. The Sun Ray system, which uses a low-level pixel-based encoding, performs as well as Metaframe or Terminal Services at high bandwidths, and provides a much higher resolution and color depth at the same time. And systems with higher-level encodings like Metaframe do not perform well at low bandwidths despite the greater efficiency of their protocols. VNC further demonstrates that a pixel-based encoding can be comparable in efficiency to a graphics-based encoding. Low-level encodings have many potential side benefits, such as platform independence and more lightweight clients, and this is an important indication that such encodings are feasible.

We also determined that while adaptive updates can be very useful in handling low-bandwidth network conditions, they can also cause rapid performance degradation for multimedia applications. This is particularly true for animation and video, where dropped frames have a greater impact on performance than increased delays, as seen in our Citrix Win2K and VNC results.

Of the individual platforms, LapLink is clearly the worst, due to a prohibitively high server overhead and inefficient packetization. The LapLink server was completely loaded during operation, and large numbers of small packets were constantly being sent. Improvements in bandwidth will be unlikely to improve this platform's performance significantly, since the bandwidth is not the bottleneck in this case. Citrix had the best overall performance of the platforms, functioning reasonably well even at 1.5 Mbps when most of the other systems began to suffer. While VNC does have an efficient protocol and maintains download speed and frame rate even at low bandwidth, this does not lead to subjectively good performance in many cases.

5 Related Work

Despite the growing interest in thin-client computing and the plethora of thin-client platforms that have been designed, quantitative performance data on thin-client platforms is quite limited. Little previous work has been done to compare the performance of thin-client platforms against one another in supporting web-based and multimedia-oriented applications. Thin-client platform vendors such as Citrix and Microsoft have conducted internal performance testing of their products, but it is unclear how much comparative evaluation they have done, especially for web and multimedia application performance. Furthermore, the product literature available from Citrix and Microsoft claim that their thin-client platforms operate well at even 56K modem speeds. Our results indicate that these claims are greatly overstated. Microsoft has also published several white papers on Terminal Server that discuss its performance for purposes of capacity planning [14, 26]. These scalability studies of thin-client computing are complementary to our work.

Several studies have examined the performance of a single thin-client system. Danskin conducted an early study of the X protocol [5]. Wong and Seltzer have also studied the performance of Windows NT Terminal Server, focusing on office productivity tools and web browsing performance [27]. Tolly Research has conducted similar studies for Citrix Metaframe [24]. Schmidt, Lam, and Northcutt examined the performance of the Sun Ray platform in comparison to the X protocol [19] and reported simulated results for Sun Ray at different network bandwidths. Our work systematically extends these studies by offering a comparative viewpoint across all of these platforms on web-based applications over a wide range of network bandwidths.

Howard presented performance results for various hardware thin-clients based on tests from the i-Bench benchmark suite [11]. This work suffers from two significant problems in measurement methodology. First, the experiments only measure benchmark performance at the server-side. They do not measure data transferred at the client-side and do not account for actual client-side performance. Second, the work was based on Microsoft Internet Explorer 5.01, which does not properly interpret the Javascript OnLoad command used in the i-Bench web page load test. This causes successive pages to start loading before previous pages have completed loading, resulting in unpredictable measurements of total web page download latencies. Netscape Navigator 4.7 does not suffer from this problem, which is why we used this browser platform for our work. Our work addresses these measurement issues not addressed in previous work, covers a broader range of platforms, and discusses a number of the underlying factors that result in the varying performance across different thin-client systems.

6 Conclusions and Future Work

We have studied the performance of thin-client computing by quantitatively evaluating the performance of five of the most popular platforms on web and multimedia applications. Our results show

that performance varies widely across different thin client platforms, but that many of the platforms can deliver excellent performance over 10 Mbps networks. This makes thin client computing a potentially effective as well as easier-to-maintain computing solution for LAN environments. However, all of the platforms that we tested showed noticeable performance problems over networks at lower than 1.5 Mbps access bandwidths. This indicates that current thin client solutions are not likely to be viable in broadband environments for web and multimedia applications.

Future research will further probe these architectures to determine what factors control their performance, and how that performance could be improved. There is also scope for research into the effect of the underlying operating system on the thin-client performance. We are also exploring other remote display technologies that might prove to be more effective in broadband environments, in particular better pixel-encoding techniques that involve intelligent pixel caching and pattern-based pixel compression. In addition, while we have examined the effects of network access bandwidth on thin client computing performance, we have not considered other issues such as scalability and reliability, which are also important dimensions of thin client computing performance. Further research will consider these issues.

7 References

1. Boca Research, "Citrix ICA Technology Brief." Technical White Paper, Boca Raton, FL, 1999.
2. Citrix Systems, "Citrix MetaFrame 1.8 Backgrounder", Citrix White Paper, June 1998.
3. Shunra Software, The Cloud, http://www.shunra.com.
4. B. C. Cumberland, G. Carius, A. Muir, *Microsoft Windows NT Server 4.0, Terminal Server Edition: Technical Reference*, Microsoft Press, Redmond, WA, Aug. 1999.
5. J. Danskin, P. Hanrahan, "Profiling the X Protocol", *Proceedings of the SigMetrics Conference on Measurement and Modeling of Computer Systems*, 1994.
6. Desktop.com, http://www.desktop.com.
7. Etherpeek 4, AG Group, Inc., http://www.aggroup.com.
8. Futurelink, http://www.futurelink.net.
9. GraphOn GO-Global, http://www.graphon.com.
10. i-Bench version 1.01, Ziff-Davis, Inc., http://i-bench.zdnet.com.
11. B. Howard, "Thin Is Back," *PC Magazine*, 19(7), Ziff-Davis Media, New York, NY, Apr. 2000.
12. LapLink.com, Inc., "LapLink 2000 User's Guide." Bothell, WA, 1999.
13. T. W. Mathers, S. P. Genoway, *Windows NT Thin Client Solutions: Implementing Terminal Server and Citrix MetaFrame*, Macmillan Technical Publishing, Indianapolis, IN, Nov. 1998.
14. Microsoft Corporation, "Microsoft Windows NT Server 4.0, Terminal Server Edition: An Architectural Overview." Technical White Paper, Redmond, WA, 1998.
15. J. Nieh, S. J. Yang, "Measuring the Multimedia Performance of Server-Based Computing," *Proceedings of the Tenth International Workshop on Network and Operating System Support for Digital Audio and Video*, Chapel Hill, NC, June 2000.
16. Personable.com, http://www.personable.com.
17. T. Richardson, Q. Stafford-Fraser, K. R. Wood and A. Hopper, "Virtual Network Computing." *IEEE Internet Computing*, 2(1), Jan/Feb 1998.
18. R. W. Scheifler and J. Gettys, "The X Window System", *ACM Transactions on Graphics*, 5(2), Apr. 1986.
19. B. K. Schmidt, M. S. Lam and J. D. Northcutt, "The Interactive Performance of SLIM: A Stateless, Thin-Client Architecture", *Proceedings of the 17th ACM Symposium on Operating Systems Principles*, Dec. 1999.
20. B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 2nd ed., Addison-Wesley, Reading, MA, 1992.
21. A. Shaw, K. R. Burgess, J. M. Pullan, P. C. Cartwright, "Method of Displaying an Application on a Variety of Client Devices in a Client/Server Network", US Patent US6104392, Aug. 2000.
22. Sun Microsystems, Sun Ray 1 Enterprise Appliance, http://www.sun.com/products/sunray1.
23. The Santa Cruz Operation, "Tarantella Web-Enabling Software: The Adaptive Internet Protocol", A SCO Technical White Paper, Dec. 1998.
24. Tolly Research, "Thin-Client Networking: Bandwidth Consumption Using Citrix ICA," IT clarity, Feb. 2000.
25. Virtual Network Computing, http://www.uk.research.att.com/vnc.
26. Microsoft Corporation, "Windows 2000 Terminal Services Capacity Planning", Technical White Paper, Redmond, WA, 2000.
27. A. Y. Wong, M. Seltzer, "Evaluating Windows NT Terminal Server Performance," *Proceedings of the USENIX 2000 Annual Technical Conference*, San Diego, CA, June 2000.
28. S. J. Yang, J. Nieh, "Thin Is In," *PC Magazine*, 19(13), Ziff-Davis Media, New York, NY, July 2000.