

Web Browsing Performance of Wireless Thin-client Computing

S. Jae Yang
Dept. of Computer Science
Columbia University
New York, NY 10027
sy180@columbia.edu

Jason Nieh
Dept. of Computer Science
Columbia University
New York, NY 10027
nieh@cs.columbia.edu

Shilpa Krishnappa
Dept. of Computer Science
Columbia University
New York, NY 10027
sk2104@columbia.edu

Aparna Mohla
Dept. of Computer Science
Columbia University
New York, NY 10027
am2104@columbia.edu

Mahdi Sajjadpour
Dept. of Electrical Engineering
Columbia University
New York, NY 10027
ms1990@columbia.edu

ABSTRACT

Web applications are becoming increasingly popular for mobile wireless systems. However, wireless networks can have high packet loss rates, which can degrade web browsing performance on wireless systems. An alternative approach is wireless thin-client computing, in which the web browser runs on a remote thin server with a more reliable wired connection to the Internet. A mobile client then maintains a connection to the thin server to receive display updates over the lossy wireless network. To assess the viability of this thin-client approach, we compare the web browsing performance of thin clients against fat clients that run the web browser locally in lossy wireless networks. Our results show that thin clients can operate quite effectively over lossy networks. Compared to fat clients running web browsers locally, our results show surprisingly that thin clients can be faster and more resilient on web applications over lossy wireless LANs despite having to send more data over the network. We characterize and analyze different design choices in various thin-client systems and explain why these approaches can yield superior web browsing performance in lossy wireless networks.

Categories and Subject Descriptors

C.4 [Computer Systems Organization]: Performance of Systems—Reliability, Availability, and Serviceability

General Terms

Experimentation, Measurement, Performance, Reliability

Keywords

thin-client computing, web performance, wireless and mobility

1. INTRODUCTION

The popularity of web applications and the increasing availability of wireless networks are fueling a growing proliferation of wireless and mobile computing devices. Furthermore, wireless thin-

client computing systems are also beginning to emerge. 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. Examples of popular thin-client platforms include Citrix MetaFrame [11, 22], Virtual Network Computing (VNC) [27] and Sun Ray [30, 34]. The remote server typically runs a standard server operating system and is used for executing all application logic.

Because thin-client systems run the program logic on faster remote servers, the client only needs to be able to display and manipulate the user interface. Clients can then be simpler devices, reducing energy consumption and extending battery life, which is often the primary constraint on the benefits of untethered Internet access with wireless devices. Thin-client users can access applications with heavy resource requirements that are prohibitive for typical mobile systems with low processing power.

While wireless thin client computing offers potential benefits, wireless networks can have high packet loss rates, which may degrade the performance of thin clients that require network access to a server at all times. For instance, packet loss in Wi-Fi (IEEE 802.11) networks [3, 14] can result from being out of range of access points or interference from physical impediments or other electronic devices. Furthermore, when it comes to web applications, the common belief is that web content should be delivered directly to a web browser running locally on the client to achieve the best web browsing performance rather than running the web browser on a remote server and relaying the web browser's display through a thin-client interface via a remote display protocol.

In this paper, we challenge this conventional wisdom. We explore the possibility that a remote thin server acting as a proxy web client running a web browser with a superior connection to the web server can allow a thin client to deliver a better web browsing experience than a mobile fat client on a wireless network that may have less than ideal network conditions. We use the term fat client to refer to systems that execute applications such as a web browser using local client computing resources. Figure 1 contrasts the op-

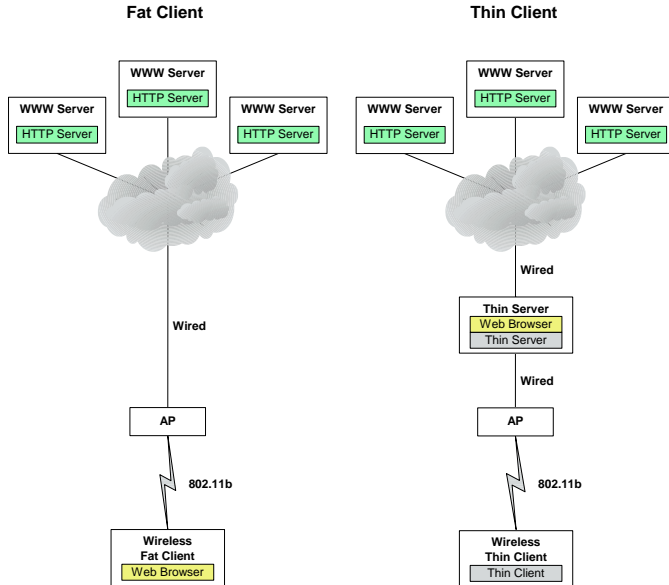


Figure 1: Wireless fat client vs. wireless thin client.

eration of a wireless fat client versus a wireless thin client in the context of web browsing.

Because the importance of thin-client computing will only continue to increase with the growing proliferation of wireless client devices, it is crucial to determine the effectiveness of thin-client computing in wireless networks on the kind of web applications that users are already using and will increasingly be using in the future. To assess the limits of wireless thin-client computing over lossy wireless networks, we have characterized the design choices of underlying remote display technologies and network protocols used and measured the impact of these choices on the performance of thin-client systems in Wi-Fi network environments, specifically IEEE 802.11b [3, 14]. For our study, we considered a diversity of design choices as exhibited by three popular thin-client systems in use today: Citrix MetaFrame, VNC, and Sun Ray. These platforms were chosen for their popularity, performance, and diverse design approaches. We conducted our experiments based on 802.11b because of its widespread use for wireless LANs. We focus on evaluating these thin-client platforms with respect to their performance on web browsing because of the importance of web applications as characterized by their widespread use. Previous studies show that web traffic constitutes the overwhelming majority of traffic in wireless networks [6, 17, 36].

We identified and isolated the impact of lossy wireless LAN environments by quantifying and comparing the performance of thin-client systems against fat clients that run web applications directly on the client. We consider the performance of these systems for a wide range of packet loss rates as exhibited in 802.11b networks. Because many thin-client systems are proprietary and closed-source, we obtained our results using slow-motion benchmarking [25], a non-invasive measurement technique we developed that addresses some of the fundamental difficulties in previous studies of thin-client performance. Our results show that thin clients can operate effectively over lossy networks, but that performance can vary widely. More surprisingly, our results show that thin clients can be faster and more resilient on web applications than fat clients

in wireless LANs, even though they send more data than fat clients using HTTP. We characterize and analyze design choices in various thin-client systems to explain why these approaches can yield superior web browsing performance in lossy wireless networks. Our results demonstrate that the fundamental characteristics of the remote display approach used in thin-client systems is a key factor in achieving better web browsing performance.

This paper is organized as follows. Section 2 gives an overview of our slow-motion measurement methodology and details the experimental testbed and application benchmarks we used for our study. Section 3 describes the measurements we obtained on both fat-client and thin-client systems in lossy network environments. Section 4 provides an interpretation of the experimental results and examines how they relate to the use of different remote display mechanisms in thin-client systems. Section 5 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 the web browsing performance of thin-client systems against traditional fat clients to assess their performance and utility in 802.11b environments. To explore a range of different design approaches, we considered three popular thin-client platforms: Citrix MetaFrame 1.8 for Windows 2000, VNC v3.3.2 for Linux, and Sun Ray I with Sun Ray Server Software 1.2.10.d. In this paper, we also refer to Citrix Metaframe by Citrix ICA (Independent Computing Architecture), which is the remote display protocol used by the Citrix platform. 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, caching mechanisms, supported display color depth, and transport protocol used. To evaluate their performance, we designed an experimental network testbed and various experiments to exercise each of the thin-client platforms on single-user web browsing 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 web application benchmarks used in our experiments.

2.1 Measurement Methodology

Because thin-client systems are designed and used very differently from conventional fat-client systems, quantifying and measuring their performance on web applications can be difficult. In traditional fat-client systems, applications run locally, and the processing unit is tightly coupled with the display subsystem so that the display updates resulting from executing the application are rendered synchronously. In thin-client systems, the application logic executes on a remote server machine, which sends only the display updates over a network to be rendered on the client's screen. The application processing unit is completely decoupled from the display subsystem; therefore, the display updates visible on the client side may be asynchronous with the application events.

To benchmark a conventional system, one can simply script a sequence of application events and let the system under test execute the script as fast as it can. The result will then generally correlate with the user experience. For example, Ziff-Davis Publishing's i-Bench is a benchmark that works in this manner to measure web browsing performance by downloading a Javascript-controlled series of web pages one after another. Each web page request is made as soon as the browser reports the current page is loaded. However, because of the lack of synchronicity between the application processing unit and the display subsystem in thin-client systems,

Platform	Display Encoding	Screen Updates	Compression and Caching	Max Display Depth	Transport Protocol
Citrix MetaFrame (ICA) 1.8	Low-level graphics	Server-push, lazy	RLE, RAM cache, disk cache	8-bit color	TCP/IP
VNC v3.3.2	2D draw primitives	Client-pull, lazy updates between client requests discarded	Hextile (2D RLE)	24-bit color	TCP/IP
Sun Ray I	2D draw primitives	Server-push, eager	None	24-bit color	UDP/IP

Table 1: Thin-client computing platforms.

benchmarks such as i-Bench that were designed for conventional fat-client systems cannot be used effectively to measure the user-perceived performance on thin-client systems. Running a conventional benchmark on a thin-client system may only test the remote server’s ability to execute the benchmark application while ignoring the visual experience of the user. With i-Bench, since the web browser runs on a remote server machine in thin-client systems, downloading web pages to a web browser in a rapid-fire manner may result in measuring only the remote server’s ability to download web objects and not reflect the latency involved in relaying the display of the web pages to the thin client for the user to see. Some thin-client systems may simply be overwhelmed and unable to catch up in rendering the display updates, while some thin-client systems may actively merge or even discard some display updates in an effort to synchronize the display events with the application events. Either way, the conventional benchmarks will not accurately account for the user’s visual experience, and thus report a meaningless measure of performance.

To address this problem, we previously developed slow-motion benchmarking and employ this methodology in this study to evaluate thin-client and fat-client web performance. A more in-depth discussion of this technique is presented in [25]. We provide an overview of slow-motion benchmarking here and we used it to measure web performance in lossy network environments. Slow-motion benchmarking employs two techniques to obtain accurate measurements: non-invasive monitoring of network activity and using slow-motion versions of web application benchmarks.

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 proprietary thin-client systems, our primary measurement technique was to use a packet monitor to capture the network traffic. The network traffic is a direct manifestation of system activity for fat clients running a web browser as well as thin-client systems relaying display information and user input. For example, as soon as a GET request is sent from a fat client to the web server, a packet is observed, and the network traffic will not cease until the client receives the last HTTP packet corresponding to the requested web page. In the case of thin-client systems, the thin server will begin to send packets to the thin client as soon as there is even a slight change in the display, and the packet transmission will continue until the display update is complete. From the packet records, we can extract the latency corresponding to the system activities that yielded the packets. To account for the delays caused by retransmission of packets in the lossy network, we monitored the network traffic on both the client-side and the server-side. Monitoring the packets only on just the client-side is inadequate, because with thin-client systems, the server may initiate the network activity corresponding to a display update. If the initial packet is dropped and retransmitted, there will be some delay before the client-side packet monitor sees the initial packet. The latency extracted from the packet records does not include the time from the moment the thin client receives the display update information to the moment all the pixels are rendered onto the screen. Similarly,

for the fat clients running a web browser, this measurement technique does not account for the time from the moment the client receives the web page data to the moment the page is fully rendered on screen. However, previous results indicate that the client-side display rendering time for web browsing is relatively small and does not change in any significant way as a result of different network environments [25].

We employed slow-motion versions of web application benchmarks to provide a measure of user-perceived performance based on the visual quality of display updates. Monitoring network activity provides a measure of the latency of display updates, but it does not provide a sufficient measure of the overall quality of the performance if not all of the display updates are actually reaching the client. To address this problem, we altered the web benchmark applications by introducing delays between the web pages, 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 packet capture data to obtain the latency and data transferred for each web page, and obtain overall results by taking the sum of these results. Section 2.3 describes in further detail how web application benchmarks were delayed for our experiments.

Our combined measurement techniques provide three key benefits. First, the techniques ensure that we can determine if 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. Consequently, we are able to 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 ICA and SunRay. Third, the techniques provide a common methodology for measuring and comparing the performance of both thin-client and fat-client systems.

2.2 Experimental Testbed

The isolated experimental testbed we used consisted of eight machines, at most six of which were active for any given test. The testbed consisted of two pairs of thin client/server systems, a web server, two packet monitor machines, and a network emulator machine. To ensure a level playing field, we used the same client/server hardware for all of our tests except when testing the Sun Ray platform, which only runs on Sun machines. The features of each system are summarized in Table 2. As discussed in Section 4, the slower Sun client and server hardware did not affect the lessons derived from our experiments. We used two packet monitor machines, both Micron Client Pro PC running Etherpeek 4 [4] network monitor software, to obtain the measurements for slow-

motion benchmarking. One packet monitor was positioned on the client-side and the other on the server-side.

We used a network emulator to adjust the network characteristics of our testbed between the respective client and server to emulate 802.11b network conditions. The network emulator machine was a Micron Client Pro PC with two 10/100BaseT NICs running NISTNet [1] network emulator software. While the 802.11b specification allows up to 11 Mbps network bandwidth, previous studies have indicated that 6 Mbps network bandwidth is more typical of what is achievable in practice [3]. The 802.11b networks are based on CSMA/CA that implements a MAC layer retransmission mechanism which retransmits the data packets that are not acknowledged by the receiver. However, CSMA/CA does not guarantee delivery at the MAC layer with retransmissions. The sender retransmits a missed packet for a fixed number of times so that the upper layer protocols such as TCP are not affected; therefore, it is possible to have transport layer packet losses in 802.11b network environments [8]. In live 802.11b network environments, the condition of the medium for RF signals changes constantly as moving objects or human bodies create temporary obstructions for the electromagnetic wave and as the wireless device users themselves move around respect to the access points. Previous studies have indicated that the transport layer packet loss rates in 802.11b networks can be up to almost 30% in practice [6]. Accordingly, we considered the web performance of the systems tested over a network with 6Mbps network bandwidth and packet loss rates up to 30%.

For our experiments using a traditional fat-client PC system, the network emulator was placed between the fat-client PC and the web server. In this case, the PC serves as a wireless client running a web browser, which then must download web information across a potentially lossy wireless network. For the experiments using thin-client systems, the network emulator was placed between the thin client and server with a 100 Mbps Ethernet connection between the thin-client server and the web server. In this case, the thin client must connect to the respective server over a potentially lossy wireless network, but the server has robust wired connectivity to other wired machines, including the web server. Typically, a thin-client environment would involve a server that is much faster than the client. In our testbed, we used the same class of system for both client and server, providing a conservative comparison of thin-client versus fat-client performance.

To minimize the differences due to application environment, 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. However, the Sun Ray client was set to 24-bit color, since the Sun Ray display protocol is based on a 24-bit color encoding. For each thin-client system, we used the server operating system that delivered the best performance for the given system; MetaFrame ran best on Windows, VNC ran best on UNIX/Linux, and Sun Ray runs only on Solaris. For comparison purposes with a traditional fat-client PC, we conducted experiments for the PC running both Linux and Windows to isolate any performance results that might be due to the choice of operating system.

2.3 Web Application Benchmarks

To measure the web performance of both thin-client and traditional PC systems, we used two web browsing application bench-

marks. For both benchmarks, we used Netscape Communicator 4.72 as the web browser and Apache 1.3.22-6 as the web server. To minimize differences in the systems, we used the same browser on all of the systems. We chose this version of Netscape because it is available on all the platforms in question. More recent browsers such as Internet Explorer 6, Mozilla 1.1, and Netscape 7 are unfortunately not available on all three of the operating systems used in our experiments. Unless otherwise stated, HTTP 1.0 was used for all experiments since that is what is used by Netscape Communicator 4.72. HTTP 1.0 remains the most widely used version of HTTP today [33]. We considered the use of nonpersistent as well as persistent HTTP connections in our experiments. In all cases, the web browser window was 1024x768 in size, so the region being updated was the same on each system. We first performed our measurements with Netscape web browser caches disabled for all platforms. To provide a conservative comparison between thin-client systems and the fat-client PC configurations, we then compared these results with using a fat client with the browser memory and disk caches enabled. When using the browser cache, we cleared both caches before each test run, set the memory and disk caches to 1 MB and 4 MB, respectively, and compared documents in the cache with documents on the network once per session by default. The memory cache size was the default value for Windows and the disk cache size was set large enough to accommodate the content of all web pages downloaded during our benchmark.

2.3.1 Multi-Page Test

The primary web benchmark we used was a multi-page test that downloads a sequence of web pages and measures the impact of lossy network conditions on the page download times. This benchmark was created by modifying the Web Text Page Load test from the Ziff-Davis i-Bench 1.5 benchmark suite [40] for slow-motion benchmarking. The original i-Bench web benchmark is a JavaScript-controlled load of a sequence of 54 web pages from the web benchmark server. 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. We modified the original i-Bench benchmark for slow-motion benchmarking by introducing delays of several seconds between pages using the JavaScript. The delays were sufficient in each case to ensure that each page could be received and displayed on the client completely without 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. For this benchmark, we conducted our measurements for various random packet loss rates ranging from no loss up to 30% packet loss.

We determined the page download times when the benchmark completed and also determined the packet loss rates at which the benchmark failed to completely download and display all of the web pages for each of the systems tested. In all cases, a successful benchmark run required all pages to be displayed at the client without interruptions. For thin clients, a benchmark run was considered to have failed to complete if the user session shut down or the server indicated that the client is disconnected. Similarly, the benchmark run was considered to have failed to complete on the fat-client PC when an error message was displayed in the browser or the browser stopped responding and there had been no packet retransmissions for more than five minutes, which was the Apache web server's default timeout period.

Role / Model	Hardware	OS / Window System	Software
PC Thin Client Micron Client Pro	450 MHz Intel PII 128 MB RAM 14.6 GB Disk 10/100BaseT NIC	MS Win 2000 Advanced Server Caldera OpenLinux 2.4, XFree86 3.3.6, KDE 1.1.2	Citrix ICA Win32 Client VNC Win32 3.3.3r7 Client Netscape Communicator 4.72
Sun Thin Client Sun Ray I	100 MHz Sun uSPARC IIep 8 MB RAM 10/100BaseT NIC	Sun Ray OS	N/A
Packet Monitor Micron Client Pro	450 MHz Intel PII 128 MB RAM 14.6 GB Disk 10/100BaseT NIC	MS Win 2000 Professional	WildPackets' Etherpeek 4
Web Server Micron Client Pro	450 MHz Intel PII 128 MB RAM 14.6 GB Disk 10/100BaseT NIC	Caldera OpenLinux 2.4	i-Bench 1.5 Apache 1.3.22-6
PC Thin Server Micron Client Pro	450 MHz Intel PII 128 MB RAM 14.6 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 VNC 3.3.3r2 for Linux Netscape Communicator 4.72
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 Generic 106541-08 OpenWindows 3.6.1, CDE 1.3.5	Sun Ray Server 1.2.10.d Beta Netscape Communicator 4.72
Network Emulator Micron Client Pro	450 MHz Intel PII 128 MB RAM 14.6 GB Disk 2 10/100BaseT NICs	Caldera OpenLinux 2.4	NISTNet 2.0.12

Table 2: Testbed machine configurations.

2.3.2 Single-Page Test

A second web benchmark we used was a single-page test to download a single page from the web server and measure the impact of intermittent connectivity during the page download. The web page downloaded was relatively large and complex, with the HTML container page alone being 85 KB in size and consisting of 40 images; the total data size of both text and graphics files was 155 KB. For this benchmark, we configured the network emulator to start with a lossless network and then immediately transition to a 100% packet loss rate after the HTTP GET request was issued and acknowledged. The packet loss rate was then reduced to zero after a specified period of time. The 100% packet loss rate for a period of time is used to represent an intermittent connectivity scenario.

We measured the total time to download the web page as well as the ability of each system to tolerate different periods of intermittent connectivity. The experiment examines the behavior of each system under a sudden network failure to determine the duration of network interruption that the systems under test could sustain and still complete a user-initiated web transaction. After each successful run, we increased the duration of the network dropout and repeated the experiment. In the single-page test, a successful run means the page is eventually downloaded and displayed upon the network recovery without any user intervention.

3. MEASUREMENTS

We ran the two web benchmarks on a fat-client PC and each of the three thin-client platforms and measured their resulting performance. We present an overview of the measurements obtained and provide some metrics of performance for each application benchmark. The web benchmark results are shown both in terms of latencies and the respective amounts of data transferred to illustrate both the overall user-perceived performance and the bandwidth efficiency of the thin-client systems compared to a traditional fat-client PC. These measurements provide the first quantitative performance comparisons of thin-client systems in lossy wireless LAN

environments. In the following discussion, we refer to MetaFrame on Windows as *ICAWin*, VNC on Linux as *VNCLin*, Sun Ray on Solaris as *SunRay*, and the PC fat client running Windows and Linux as *PCWin* and *PCLin*, respectively.

3.1 Multi-Page Test

Figures 2 through 8 show the measurements for running the multi-page web benchmark on the thin-client systems and the fat-client PC over the range of packet loss rates found within the coverage area of 802.11b LANs [6]. Measurements are only shown for experiments in which the benchmark completed, resulting in all of the web pages being successfully downloaded and displayed on the client.

We first ran all of the experiments with non-persistent HTTP connections and all client and web caching disabled. These measurements are shown in Figures 2 through 4. Figure 2 shows the average latency for downloading a web page at each packet loss rate for each system. Experts differ on the amount of latency considered acceptable for downloading a web page. Some usability studies have shown that web pages should take less than one second to download for the user to experience an uninterrupted browsing process [26], while others indicate that the current ad hoc industry quality goal for download times is 6 seconds [16]. With no packet loss, our measurements show that all of the systems provide good performance with subsecond average page download times. With no packet loss, the PC fat clients provided the best performance with average page download times of less than 250 ms. In comparison, the average page latencies for the thin-client systems were roughly 460 ms for *VNCLin*, 830 ms for *SunRay*, and 780 ms for *ICAWin*. The PC fat clients perform better than the thin clients at 1% packet loss as well, but some of the thin clients start to outperform some of the PC fat client configurations at 2% packet loss, where *VNCLin* delivers faster page download times than *PCWin*. Regardless of these differences, the page download times below 4% packet loss were roughly a second or less for all of the systems.

However, at higher packet loss rates, *SunRay* and *VNCLin* per-

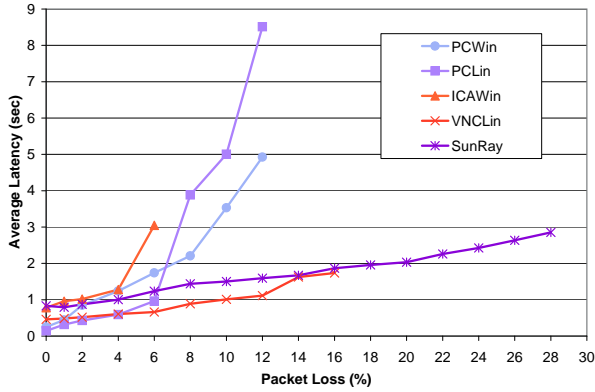


Figure 2: Average latency per page for multi-page web benchmark with no cache and non-persistent connection.

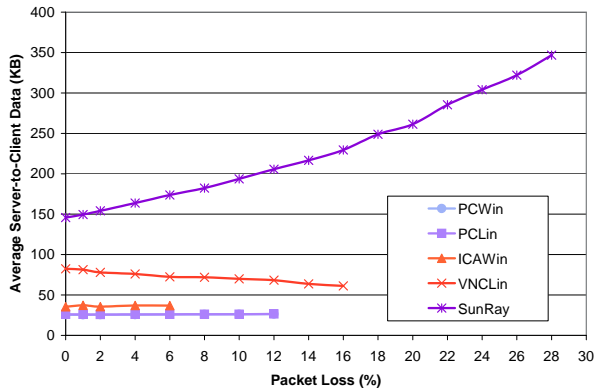


Figure 3: Average data transferred per page from server to client for multi-page web benchmark with no cache and non-persistent connection.

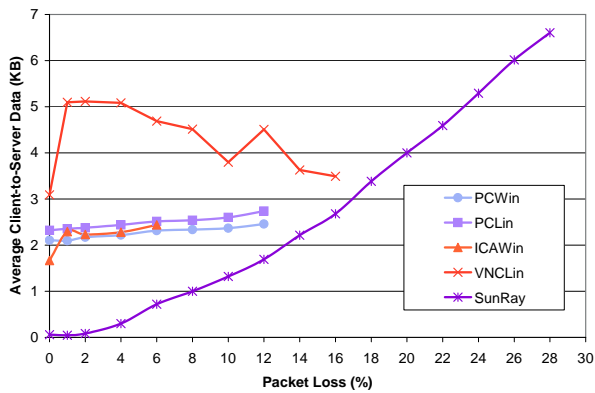


Figure 4: Average data transferred per page from client to server for multi-page web benchmark with no cache and non-persistent connection.

formed the best while the PC fat clients and ICAWin became unacceptable. ICAWin had the slowest page download times at all packet loss rates and failed to complete the benchmark for packet loss rates above 6%. While the PC fat clients continued to complete the benchmark for packet loss rates above 6%, the latencies incurred on these systems was significantly higher than for lower packet loss rates, ballooning to more than 8 seconds in some cases. More importantly, the fat clients incurred much higher average page download times than the other thin-client systems for packet loss rates above 6%. Furthermore, only the thin-client systems were able to complete the benchmark for packet loss rates above 12%. SunRay was the most resilient of all the systems, successfully completing the benchmark for packet loss rates up to 28% with only a gradual increase in page latencies.

Figure 3 shows the average amount of data transferred from server to client per web page for each system when running the multi-page benchmark. Figure 4 shows the average amount of data transferred from client to server per web page for each system when running the multi-page benchmark. As expected, substantially less data is sent from client to server than from server to client for this web benchmark. For most of the systems, the amount of data transferred from server to client and client to server remains relatively constant across changes in packet loss rate. This is expected since the slow-motion benchmarking technique we used ensures that each web page is completely downloaded and displayed on the client. However, SunRay and VNCLin veered away from this expectation. The amount of data transferred using SunRay increased monotonically as the packet loss rate increased, and the amount of data transferred from server to client using VNCLin showed a downward trend as the packet loss rate increased. We will discuss the reasons for the SunRay and VNCLin behavior in Section 4.

The measurements in Figure 3 show that SunRay transferred the most amount of data while the PC fat clients transferred the least amount of data. Among the thin-client systems, ICAWin transferred the least amount of data, sending only 40% more data than the PC fat clients. Figures 2 and 3 show that the thin-client system that sent the least amount of data had the worst performance, and the thin-client system that sent the most amount of data was the most resilient, tolerating almost twice the packet loss as the PC fat clients. Even VNCLin outperformed the PC fat client configurations at higher packet loss rates despite sending noticeably more data. Note that we also measured the number of packets that were transferred for each system and found that the relative differences among the systems was similar to those shown by the data transfer measurements in Figure 3.

Next, we ran all of the experiments again with non-persistent HTTP connections but with client caching enabled to quantify the benefits using caching at the client to improve web browsing performance in wireless networks. In particular, we enabled the web browser caches for PCWin and PCLin and enabled the caches for ICAWin. VNCLin and SunRay did not have client caching mechanisms and therefore provided the same performance as before. To provide a more conservative comparison, we did not enable the web browser caches for the thin-client systems since they reside on the server instead of the client. Note that the function of the thin-client caches used in ICAWin differed from that of the fat client web browser caches. The fat clients cache the actual web objects such as the HTML file or embedded images while ICAWin caches bitmap images or fonts that appear on screen only. For example, ICAWin will not cache a JPEG image that is placed below the visible part of the page.

Figures 5 through 7 show the measurements for running the multi-page web benchmark on the fat-client and thin-client systems with

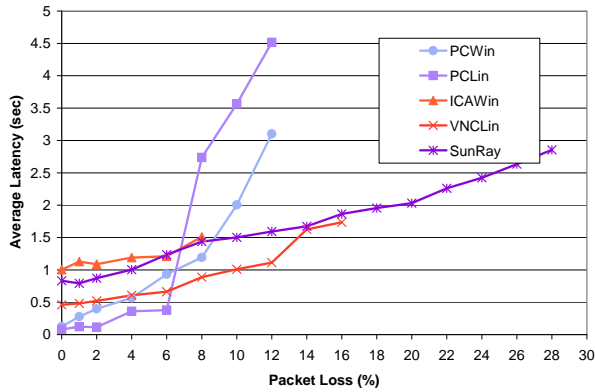


Figure 5: Average latency per page for multi-page web benchmark with cache and non-persistent connection.

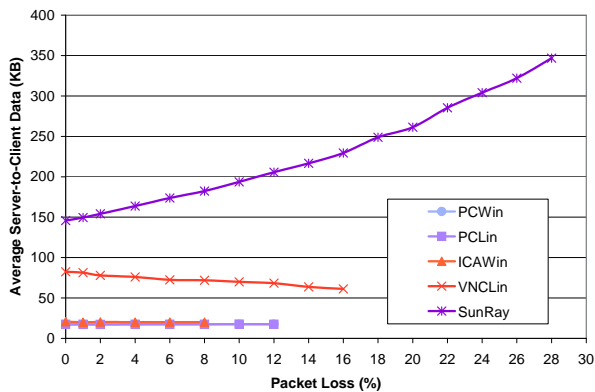


Figure 6: Average data transferred per page from server to client for multi-page web benchmark with cache and non-persistent connection.

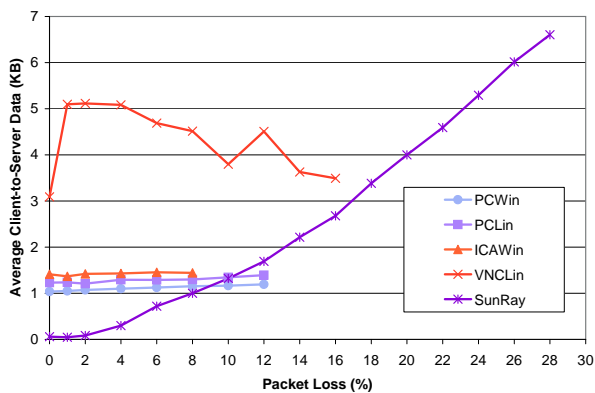


Figure 7: Average data transferred per page from client to server for multi-page web benchmark with cache and non-persistent connection.

client caching enabled. Figure 5 shows the average latency for downloading a web page at each packet loss rate. The latency measurements show that thin-client systems without caching mechanisms, SunRay and VNCLin, were more resilient and delivered faster page download times at higher packet loss rates than those systems that provided a caching mechanism. SunRay and VNCLin performed much better than the fat clients at higher packet loss rates even with web browser caching enabled for the fat clients. This was surprising since the multi-page benchmark reused a number of images from one page to another and downloaded the same set of web pages twice during the experiment. These characteristics of the benchmark would tend to favor systems that employed caching.

Figures 6 and 7 show the corresponding average amount of data transferred per page from server to client and from client to server, respectively. Figure 4 and Figure 7 show that client caching halved the amount of data sent from the client to server for the fat clients and reduced the client-to-server data for ICAWin by roughly 40%. The reduction in data size is expected as the clients need to send less requests and acknowledgement to the server. Comparing Figures 3 and 6, we can see that client caching was also effective in reducing the amount of data transferred from server to client for those systems that provided a caching mechanism. Caching was able to reduce the data transferred by roughly 35% for fat clients and by about 44% for ICAWin. Comparing Figure 5 with Figure 2, we can see that client caching significantly reduced the latency for the fat clients and for the ICAWin thin client especially at higher packet loss rates. The PC fat clients derived substantial benefit from web caching since the 54 web pages were downloaded twice. The disk cache size was sufficient to accommodate all of the data pertaining to the 54 web pages, but the data size and latency were not exactly halved, because we designed the benchmark such that the HTML files still must be downloaded during the second iteration. Only the graphics objects can yield cache hits. As expected, the effect of caching in terms of latency was more pronounced as the packet loss rate increased. Despite the improvements in latency, web caching did not result in any improvement in resiliency in the presence of packet loss for the fat clients. ICAWin, however, showed slightly better resiliency when the cache was engaged.

Because persistent HTTP connections can improve web browsing performance, we ran all of the experiments with the fat clients again with persistent HTTP connections and client web browser caching enabled to quantify the benefits of using persistent HTTP connections to improve web browsing performance in wireless networks. Figure 8 shows the average latency for downloading a web page at each packet loss rate, where the fat clients PCWin and PCLin are using persistent HTTP 1.0 connections. The measurements for thin-client systems are the same as those shown in Figure 5 with client caching enabled and non-persistent HTTP connections over the wired network between the respective thin-client server and the web server. The measurements show that the fat clients performed much better using persistent HTTP connections under all packet loss conditions versus using non-persistent HTTP connections. The improvement was amplified as the packet loss rate increased. However, the resiliency of the fat clients did not improve with PCWin and PCLin failing to complete the benchmark when the packet loss rate reached above 12%. As in previous cases, the latency measurements show that SunRay and VNCLin were still more resilient and delivered faster page download times at higher packet loss rates than PC fat clients using persistent HTTP. As an additional comparison, we also ran the multi-page benchmark using PCWin and PCLin with Mozilla 1.1 which provides support for persistent HTTP 1.1. The resulting measurements showed that per-

sistent HTTP 1.1 did not provide any further performance improvement, still delivering worse performance than SunRay and VNCLin at higher packet loss rates.

3.2 Single-Page Test

Figure 9 shows the measurements for running the single-page web benchmark on the thin-client systems and the fat-client PC under intermittent connectivity conditions in which the network suffers 100% packet loss for a fixed time duration and is then restored. Figure 9 shows the latency for downloading the single web page in the presence of a range of network dropout time durations from no dropout to a dropout duration of more than 20 minutes. Measurements are only shown for experiments in which the web page was successfully downloaded and displayed. Enabling and disabling caching had no measurable impact for this experiment for the systems that supported caching, so results are only shown for those systems with caching enabled.

The measurements in Figure 9 show that the web page takes roughly 5 seconds to be downloaded and displayed on the client when there is no network dropout for all of the systems used. The PC fat clients PCLin and PCWin had slightly lower page download latencies in the absence of any packet loss, but all of the systems provided reasonable performance, downloading and displaying the web page on the client in less than 6 seconds. However, as the dropout duration increases, the performance of the different systems diverges. ICAWin has the worst performance as all systems both in terms of page latency as well as resiliency. For network dropouts of longer than 13 seconds, ICAWin fails to complete the benchmark and instead completely drops the user session. In comparing the other systems, the measurements show that as the network dropout duration increases, the page latencies incurred by PCLin and PCWin grow at a faster rate than those incurred by the VNCLin and SunRay thin clients as the thin-client systems deliver better page latencies at higher dropout durations than the fat clients. Furthermore, while PCLin and PCWin fail to download the page for dropout durations of more than 220 seconds, VNCLin and SunRay are much more resilient as they are able to recover from much longer dropout durations and successfully display the web page. VNCLin is able to successfully complete the benchmark even at a network dropout duration of 800 seconds and SunRay was able to successfully complete the benchmark at all dropout durations that we tested, including 1280 seconds, which was the highest duration we tested.

Figure 10 shows the amount of data transferred during the single page download. There are two observations worth noting. First, two of the thin-client systems, ICAWin and VNCLin, sent less data than the PC clients. For this benchmark, the remote display encodings used by ICAWin and VNCLin were more efficient in delivering the display of the web page to the client than native HTML. Since the remote display mechanisms only display what is actually viewed on the client, they do not need to display portions of the web page that are not displayed whereas the PC fat clients need to completely download all of the data associated with the web page. Second, the performance of the thin-client systems was inversely related to the amount of data that each system sent. The system that sent the least amount of data across the network between server and client, ICAWin, was the one that had the worst page latency and was the least resilient in the presence of a network dropout. On the other hand, SunRay was the most resilient system despite also transferring the most data of all the systems tested.

4. INTERPRETATION OF RESULTS

Our measurements show that thin-client systems, depending on

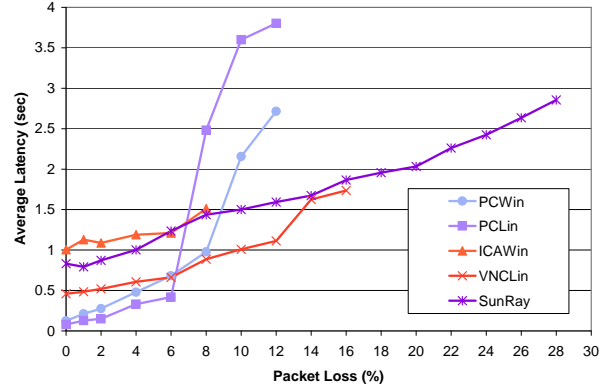


Figure 8: Average latency per page for multi-page web benchmark with cache and persistent connection.

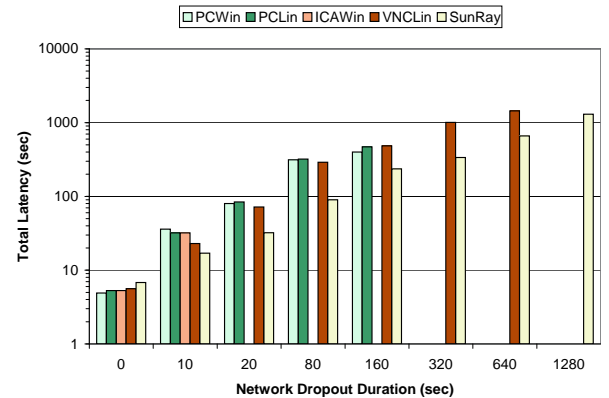


Figure 9: The recovery time at various durations of network dropout during a single-page download.

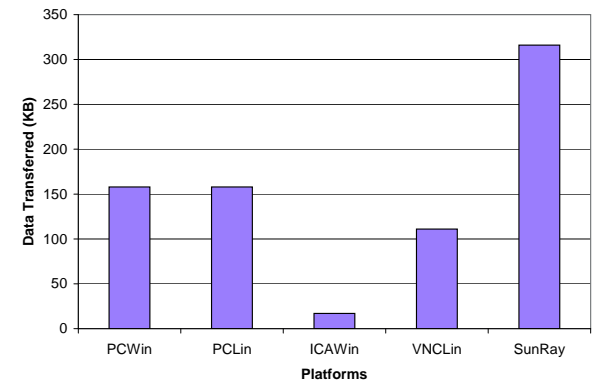


Figure 10: The data transferred from the server to client during the single-page download test.

their design and implementation, can provide faster web page download latencies and more resilient web browsing behavior than fat-client PCs in the presence of higher packet loss rates in wireless networks. These results are counterintuitive given that our measurements also show that these thin-client systems generally transmit more data during operation than the fat-client PCs. One would typically expect that as packet loss rates increase, a system that sends less data would provide better performance than a system that sends more data. At the same time, while thin-client systems VNCLin and SunRay outperformed PCLin and PCWin fat clients at higher packet loss rates, thin-client system ICAWin fared worse than the other systems. To explain the reasons for the behavior shown in our measurements, we focus on the transport protocols used for each system and the computing model afforded by thin clients versus fat clients. We discuss the impact of four different issues: TCP retransmission frequency, number of TCP connections used, models of client-server interaction, and thin-client display update merging.

4.1 TCP Retransmission Frequency

As shown in Table 1, all of the thin clients were based on TCP except for SunRay, which uses UDP. In addition, the fat clients use HTTP which is built on top of TCP. Although all of the systems measured used TCP except for SunRay, our results show that the TCP-based systems provided varying levels of performance. Given that the bulk of the data sent for all systems was from server to client, the main factor that accounts for the performance difference between ICAWin and the other TCP-based systems is the TCP implementation of the server operating system. ICAWin was the only system that we tested with a Windows 2000 server operating system. All of the other TCP-based systems used a Linux server, with the fat clients connecting to a web server running Linux and VNCLin using a thin server running Linux.

TCP provides a reliable data connection in the presence of packet loss by retransmitting packets that have not been acknowledged within a specified timeout interval. Typically the timeout interval is doubled for each retry providing exponential backoff behavior. However, the number of retries that occur before a connection is closed is determined by the TCP implementation. For an established TCP connection, Windows 2000 retransmits a failed packet up to 5 times, doubling the timeout period after each failure [21]. For the Linux systems, we used the default TCP retransmission setting which calls for retransmitting a failed packet up to 15 times, doubling the timeout period after each failure until the timeout period reaches 2 minutes [2]. In other words, Linux tries more times than Windows to get a packet delivered in the presence of loss. Fewer retransmission attempts makes Windows-based systems less resilient under rising packet loss rate.

As a result, ICAWin ends up being the least resilient system because it uses a Windows server operating system. For the multi-page test, it failed to complete the benchmark for packet loss rates beyond 8%. For the single-page test, it only survived network dropouts of up to 13 seconds. While the network is disconnected, packets will be dropped and the Windows TCP implementation will retransmit a dropped packet up to five times. With a network dropout duration of 13 seconds, it turns out that the fifth retry occurs after the dropout duration and would successfully reach the ICAWin client and start delivering the contents of the web page. With a longer network dropout duration, the fifth retry gets dropped, causing the Windows server to timeout and close the connection.

4.2 Number of TCP Connections Used

A primary factor that accounts for the performance difference

between VNCLin and the PC fat clients is the difference in the way thin-client and fat-client systems use TCP connections for web browsing. In particular, while TCP retransmits dropped packets using an increasing timeout interval providing exponential backoff behavior, the timeout interval TCP uses depends on the type of packet being sent. TCP distinguishes between SYN packets that are used to establish a TCP connection versus packets transmitted over an already established TCP connection. When retransmitting a dropped SYN packet, TCP retries the first time after an initial timeout of 3 seconds. When retransmitting a packet over an already established connection, TCP adjusts the initial timeout interval based on the round-trip time (RTT) between the two endpoints of the connection as calculated by TCP, which is typically much smaller than 3 seconds. For example, in our testbed, the TCP implementation in Linux would set the initial timeout interval for retries for an established connection to be as small as 200 ms. As a result, the initial timeout interval for SYN packets is much longer than packets for an established connection and successive retries of SYN packets will take much longer to occur compared to retries for an established connection.

VNCLin establishes one TCP connection through which all display updates are transmitted from server to client. On the other hand, PC fat clients that communicate with a web server via HTTP may open many connections in order to download a series of web pages. For the multi-page benchmark, the fat-client systems opened about 290 TCP connections when using non-persistent HTTP 1.0 and roughly 130 TCP connections when using persistent HTTP 1.0. At higher packet loss rates, the SYN packets used to establish new connections will need to be retransmitted more often and incur long delays due to the relatively large timeout interval used. The more connections that need to be opened, the greater the delay that will occur due to SYN packet dropping. Since VNCLin only opens one TCP connection over the wireless network and maintains it throughout the benchmark run, VNCLin performed much better than the PC clients on the multi-page test at higher packet loss rates.

One may argue that the advantage of using a single connection as seen in VNCLin can also be leveraged by using persistent HTTP connections for the fat clients. However, even with persistent HTTP, our results show that many connections may still be opened even when downloading web content from a single web server. Even if all objects come from one source, a typical Web server such as Apache limits the number of maximum GET requests handled per each persistent session. In the Apache server we used, the default maximum value was 100. Also, by default, if two consecutive GET requests are over 15 seconds apart, the persistent session resets, requiring a fresh connection. In practice, the objects on a web page or a sequence of web pages may come from many different web servers, requiring a new connection for each object.

4.3 Models of Client-server Interaction

VNCLin is more resilient than the fat clients on the single-page test for a related but somewhat different reason. For the single-page test, all clients had already established a TCP connection with their respective servers and were awaiting replies from the server before the network was transitioned to a 100% packet loss rate. Since VNCLin, PCWin, and PCLin all connected to a Linux server, the TCP implementation of the server for these three cases was the same. For our experiments, Linux TCP sends the first retry after 200 ms with successive retries based on exponential backoff behavior. Linux allows up to fifteen retries for an established TCP connection and sets the maximum interval between retries to two

minutes. For fifteen retries, the last few retries will occur two minutes apart instead of at exponentially increasing intervals. For the single-page test, VNCLin survives network dropouts of up to 800 seconds. With a network dropout duration of 800 seconds, the fifteenth retry occurs after the dropout duration and would successfully reach the VNCLin client and start delivering the contents of the web page. With a longer network dropout duration, the fifteenth retry gets dropped, causing the VNC server to timeout and close the connection.

Since PCLin and PCWin use the same Linux server, one might expect that they would also survive network dropouts of the same duration as VNCLin. However, our measurements show that PCLin and PCWin only survive network dropouts of up to 220 seconds. The reason for this is that the Apache web server has a default timeout setting of 300 seconds. When Apache tries to respond to the GET request, it keeps trying to retransmit the data to the client based on TCP exponential backoff behavior. It sends the first retry after roughly 200 ms and sends another nine retries, the tenth occurring roughly 110 seconds after its previous retry and roughly 220 seconds after the first retry. With a network dropout duration of 220 seconds, the tenth retry would occur after the dropout duration and would successfully reach the PC client and start delivering the contents of the web page. With a longer network dropout duration, the tenth retry would get dropped. The next retry would not occur until roughly 220 seconds later, roughly 440 seconds after the first retry. Since the Apache web server has a default connection timeout value of 300 seconds, Apache closes the connection before the next retry, limiting the dropout duration that can be tolerated by PCLin and PCWin.

One may argue that a larger timeout value can be used for Apache to provide fat clients with comparable performance to VNCLin on the single-page test. However, there is a fundamental difference in the thin-client model versus the fat-client model for web browsing. In the thin-client model, the client is expected to connect to a single server for an extended period of time and hence using the default TCP values for the number of retries make sense. In the fat-client model for web browsing, the client is expected to jump frequently from one web server to another with the amount of time spent at any given web server typically being short. A busy web server may need to handle many connections since web browsers typically open multiple connections to download a given web page and may not close their connections even after a page download has completed. As a result, web servers such as Apache provide shorter timeout values to avoid having to maintain extra resources associated with connections that are no longer needed.

Unlike the TCP-based systems, SunRay was able to complete the single-page benchmark successfully for all network dropout durations. Its mechanism allows it to automatically re-establish a connection between client and server for any dropout duration. Unlike HTTP servers or TCP-based systems, SunRay assumes a model in which clients are assumed to persistently connect unless they explicitly disconnect. Each session is a single UDP stream instead of multiple TCP connections as in the case of HTTP, which reduces the amount of state that SunRay needs to maintain per session. By leveraging an explicit thin-client approach, SunRay can more efficiently support a persistent connection that improves resilience in lossy wireless networks than HTTP or other TCP-based systems.

4.4 Thin-client Display Update Merging

VNCLin also provides additional benefit due to its thin-client computing model. Using a thin client, only the screen display updates are being sent over the lossy wireless network, not the web transactions themselves. For screen updates, what matters is for

the user to be able to view the current contents of the display. VNC takes advantage of this property by only sending display updates when requested by the client and merging display information on the server. If there are multiple updates to the same region of the display between client update requests, display merging overwrites the display region with the latest display update so that only the latest display information needs to be sent to the client when it is requested. The VNC client only requests new display updates when it has completed processing previous updates. In the presence of increased packet loss, there is an increased delay in transmitting the screen updates to the client, resulting in an increased delay in the client processing those updates. This reduces the frequency at which the client will issue requests for new display updates. Since the server has more time between sending display updates to the client, it ends up rendering the web pages almost in their entirety between client update requests so that the client ends up just seeing the web pages completely displayed as opposed to the intermediate results. Because intermediate results are skipped more often with increased packet loss, VNC ends up sending less data as packet loss rates increased even though it completely displays the web pages on the VNC client.

Since all the processing of the web pages occurs on the server, the packet loss rates between VNC client and server do not interfere with any web transactions. Furthermore, even if the connection from the wireless client to the server fails due to packet loss, any pending web transactions would still complete on the thin-client server since it has a reliable wired connection to the web server. As a result, with a thin-client computing model, when wireless connectivity is restored at a later time, a user can simply reconnect from the wireless client to the thin-client server to view the successfully completed web transaction. With a PC fat client, the user would be forced to retry the web transaction instead.

SunRay takes further advantage of the characteristics of the thin-client computing model by using a UDP-based protocol instead of using TCP. One key difference between UDP and TCP is that TCP requires acknowledgment for the packets transmitted while UDP does not. Using a reliable transport protocol such as TCP, display data lost due to packet loss is retransmitted. Since a wireless thin client is only receiving display updates over a lossy wireless network, there is no reason to retransmit old display data if newer display updates are available. By using UDP, SunRay does not have to retransmit old display data that is lost if newer display updates are available. SunRay assigns sequence numbers to each display command that is sent from the server to the client. If packet loss results in missing data in a display command received by the client, the client requests that the server resend the display information corresponding to the region of the screen that is modified by the missing display command. If a more recent display command occurs that updates the given region of the screen, the server sends the new display command instead of resending the old display command. By always sending the latest display information, SunRay is able to do a better job of hiding delays that occur due to packet loss. Because of this feature, the SunRay client sends more data to the server as the packet loss rate increases because it must request the server to resend more display information. The granularity at which the server resends display information is a display command, so dropping a single packet can result in several packets being resent when the display command requires multiple packets. This results in more data being sent from server to client as the packet loss rate grows. By leveraging the characteristics of display updates in the thin-client computing model, SunRay is able to provide better page latencies and more resiliency in lossy wireless networks even while sending more data than the fat-client PC approaches.

5. RELATED WORK

In addition to the popular thin-client systems discussed in this paper, many other systems for remote display have been developed. These include Microsoft Terminal Services [12, 23], Tarantella [28, 32], X [29] and extensions such as low-bandwidth X (LBX) [9], Kaplinsk's VNC tight encoding [15], as well as remote PC solutions such as Laplink [20] and PC Anywhere [35]. Several studies have examined the performance of thin-client systems [5, 13, 18, 30, 37, 24, 38, 25, 39]. These studies have focused on measuring the performance of thin clients in network environments with different network bandwidths and latencies, but have not considered the impact of packet loss rates in wireless networks on thin-client performance.

IEEE 802.11b wireless LANs are increasingly common, but there have not yet been many studies that model and characterize these network environments. Most studies that have been done to examine wireless network usage and analyze overall user behavior have focused on university campus networks [6, 17, 36]. These studies indicate that the majority of network traffic is due to web traffic. One study measuring an 802.11b network used for an ACM conference also measures packet loss in these environments and finds loss rates average less than 4% but can be as high as 28% [6]. As a result of these studies, our work has focused on web browsing performance over lossy wireless networks.

Several studies have considered TCP performance in wireless networks and in the presence of higher packet loss rates, including [7, 19, 31]. These studies suggest that reliable transport protocols such as TCP do not perform as well on wireless networks. In wireless networks, packet loss is often caused due to other factors besides congestion. Wireless channels often suffer from highly variable bit error rates and intermittent connectivity. TCP unfortunately assumes that these losses are due to congestion and invokes its congestion control measures. This results in an unnecessary reduction in end-to-end throughput and thus sub-optimal performance.

Given the dominance of HTTP traffic, much work has been done to measure and improve the performance of HTTP, including improving the interactions between HTTP and TCP. While recent studies indicate that HTTP 1.0 is still used far more frequently than HTTP 1.1 in practice [33], previous work indicates that HTTP 1.1 can deliver better performance overall, including in wireless network environments [10]. However, our results show that a wireless thin-client computing approach can yield better performance on web browsing applications using either HTTP 1.0 or HTTP 1.1 with traditional fat-client PCs.

6. CONCLUSIONS AND FUTURE WORK

In measuring and comparing the web browsing performance of thin-client systems against the traditional HTTP clients under wireless network conditions of various quality, we have demonstrated the conventional wisdom that web browsing should be done on a locally running browser does not apply under degraded network conditions. Our findings indicate that the conventional HTTP clients have the performance advantage when the network conditions are close to ideal, but when the wireless client encounters a degraded network quality, browsing the web through a thin client ensures more reliable web transactions and faster response time.

We produced the supporting data by employing the web benchmark implemented with the slow-motion benchmarking technique that ensures a fair comparison between fat and thin clients. We found that as the packet loss rate increases in a degrading network, the error correction behavior significantly impacts the web browsing performance. Most of the thin-client platforms and the tradi-

tional HTTP clients depend on the TCP to handle the errors. But while a thin client maintains one connection with the thin server during a user session, a fat client running a local browser must establish multiple TCP connections as the user browses multiple locations on the web. As the fat client makes a new connection, any packet error encountered introduces a significantly longer delay than it does for a thin client. SunRay mitigates the effect of packet losses even further by employing a more sophisticated error correction mechanism over UDP rather than relying on TCP.

In this study, we have studied the thin-client and fat-client performance under wireless network conditions corresponding to 802.11b. As a variety of wireless networks become popular, and as more mobile devices run client-server applications via the World Wide Web, the thin-client web performance at diverse wireless network environments and a wide range of mobile devices merits further studies in the future.

7. ACKNOWLEDGMENTS

This work was supported in part by NSF grants EIA-0071954 and CCR-0219943. We thank Jonathan So who assisted with running the benchmark tests and Albert Lai for his insightful comments on our earlier revisions.

8. REFERENCES

- [1] National Institute of Standards and Technology. <http://snad.ncsl.nist.gov/itg/nistnet/>.
- [2] TCP Manual Page. Linux Programmer's Manual, 1999.
- [3] 80211 planet. <http://www.80211-planet.com/>.
- [4] Etherpeek 4. <http://www.aggroup.com>.
- [5] C. Aksoy and S. Helal. Optimizing Thin Clients for Wireless Active-media Applications. In *Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'00)*, Monterey, CA, 2000.
- [6] A. Balachandran, G. Voelker, P. Bahl, and V. Rangan. Characterizing User Behavior and Network Performance in a Public Wireless Lan. In *ACM SIGMETRICS'02, Marina Del Rey, CA*, June 2002.
- [7] H. Balakrishnan and R. H. Katz. Explicit Loss Notification and Wireless Web Performance. Nov. 1998.
- [8] P. Brenner. *A Technical Tutorial on the IEEE 802.11 Protocol*. BreezeCom Wireless Communications, 1997.
- [9] Broadway / X Web FAQ. <http://www.broadwayinfo.com/bwfaq.htm>.
- [10] S. Cheng, K. Lai, and M. Baker. Analysis of HTTP/1.1 Performance on a Wireless Network. Technical report, Department of Computer Science, Stanford University, 1999.
- [11] Citrix MetaFrame 1.8 Backgrounder. Citrix White Paper, Citrix Systems, June 1998.
- [12] B. C. Cumberland, G. Carius, and A. Muir. NT Server 4.0, Terminal Server Edition: Technical Reference, Aug. 1999.
- [13] J. Danskin and P. Hanrahan. Profiling the X Protocol. In *ACM SIGMETRICS Performance Evaluation Review*, Nashville, TN, May 1994.
- [14] M. S. Gast. *802.11 Wireless Networks, The Definitive Guide*. O'Reilly and Associates, Sebastopol, CA, Apr. 2002.
- [15] C. Kaplinsk. Tight encoding. <http://www.tightvnc.com/compare.html>.
- [16] T. Keeley. Thin, High Performance Computing over the Internet. In *Eighth International Symposium on Modeling, Analysis and Simulation of Computer and*

- Telecommunication Systems MASCOTS'2000, San Francisco, CA, San Francisco, CA, Aug. 2000.*
- [17] D. Kotz and K. Essien. Analysis of a Campus-wide Wireless Network. In *Eighth Annual International Conference on Mobile Computing and Networking*, Sept. 2002.
- [18] A. Lai and J. Nieh. Limits of Wide-Area Thin-Client Computing. In *ACM SIGMETRICS'02*, Marina Del Rey, CA, June 2002.
- [19] T. V. Lakshman and U. Madhow. The Performance of TCP/IP for networks with High Bandwidth-Delay Products and Random Loss. In *IEEE/ACM Transaction on Networking*, 1997.
- [20] *LapLink 2000 User's Guide*. Bothell, WA, 1999.
- [21] D. MacDonald and W. Barkley. Microsoft windows 2000 tcp/ip implementation details. Technical report, 2000.
- [22] 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.
- [23] Microsoft Windows NT Server 4.0, Terminal Server Edition: An Architectural Overview, 1998.
- [24] Windows 2000 Terminal Services Capacity Planning, 2000.
- [25] J. Nieh, S. J. Yang, and N. Novik. Measuring Thin-Client Performance Using Slow-Motion Benchmarking. In *ACM Transactions on Computer Systems (TOCS)*, 21(1), Feb. 2003.
- [26] J. Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, 2000.
- [27] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1), Jan./Feb. 1998.
- [28] Tarantella Web-Enabling Software: The Adaptive Internet Protocol, Dec. 1998.
- [29] R. W. Scheifler and J. Gettys. The X window system. In *ACM Transactions on Graphics*, volume 5, 1986.
- [30] 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, Kiawah Island Resort, SC, Kiawah Island Resort, SC, Dec. 1999.*
- [31] A. Shah. TCP Performance over Wireless Links, EE359 Course Notes, Stanford University, Dec. 2001.
- [32] 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.
- [33] F. Smith, F. Campos, K. Jeffay, and D. Ott. What TCP/IP Protocol Headers Can Tell Us About The Web. In *ACM SIGMETRICS Performance Evaluation Review*, volume 29, pages 245–256, 2001.
- [34] Sun Ray 1 Enterprise Appliance. <http://www.sun.com/products/sunray1>.
- [35] PC Anywhere. <http://www.symantec.com/pccanywhere>.
- [36] D. Tang and M. Baker. Analysis of a Local-Area Wireless Network. In *Mobicom 2000*, Boston, MA, Aug. 2000.
- [37] Thin-Client Networking: Bandwidth Consumption Using Citrix ICA. *IT clarity*, Feb. 2000.
- [38] 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.
- [39] S. J. Yang, J. Nieh, M. Selsky, and N. Tiwari. The Performance of Remote Display Mechanisms for Thin-Client Computing. In *2002 USENIX Annual Technical Conference*, Monterey, CA, June 2002.
- [40] i-Bench version 1.5. <http://i-bench.zdnet.com>.