Improving Web Browsing on Wireless PDAs Using Thin-Client Computing

Albert M. Lai, Jason Nieh, Bhagyashree Bohra,
Vijayarka Nandikonda, Abhishek P. Surana, and Suchita Varshneya

Department of Computer Science
Columbia University
New York, NY 10027
{amlai, nieh, bn111, vn2107, aps2105, sv2118}@cs.columbia.edu

ABSTRACT

Web applications are becoming increasingly popular for mobile wireless PDAs. However, web browsing on these systems can be quite slow. An alternative approach is handheld thin-client computing, in which the web browser and associated application logic run on a server, which then sends simple screen updates to the PDA for display. 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 on a PDA. Our results show that thin clients can provide better web browsing performance compared to fat clients, both in terms of speed and ability to correctly display web content. Surprisingly, thin clients are faster even when 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 on mobile wireless PDAs.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication-Networks**]: Distributed Systems; C.4 [**Performance of Systems**]: Measurement techniques

General Terms

Experimentation, Measurement, Performance

Keywords

thin-client computing, web performance, wireless and mobility

1. INTRODUCTION

The increasing ubiquity and decreasing cost of Wi-Fi is fueling a proliferation of wireless PDAs (Personal Digital Assistants). These devices are enabling new forms of mobile computing and communication. Organizations are beginning to use these wireless networks and devices to deliver real-time access to web-enabled information to end users. This is typically done by running a web browser on the PDA to provide access to web applications.

An alternative approach to deliver web-enabled information using thin-client computing. 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 [7, 20], Microsoft Terminal Services [8, 21], AT&T Virtual Network Computing (VNC) [27], and Tarantella [28, 31]. The remote server typically runs a standard server operating system and is used for executing all application logic.

Figure 1 shows the traditional web browsing model. We refer to this model with the term fat client because all application logic executes on the web browser on the client device. Figure 2 in contrast shows the thin-client computing model. In the thin-client case, the web browser runs on the thin-client server instead of the client device. Only a simple thin-client application for processing user input and screen updates needs to run on the client.

With thin-client computing, because all application processing is done on the server, the client only needs to be able to display and manipulate the user interface. It does not need to run a complex web browser. 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 PDAs. Thin-client users can access applications with heavy resource requirements that are prohibitive for typical mobile systems with low processing power. Furthermore, because the client in the thin-client model does not run application logic, it does not maintain application state. This provides for much better information security because if an insecure PDA is lost or stolen, no sensitive application state is available on the device.

Despite the potential benefits of thin-client computing, an important issue in the context of web applications is to understand what kind of web browsing performance a thin-client approach provides. 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 fact, previous work [38] in a simulated lossy Wi-Fi environment suggests that while thin clients may perform better than fat clients under very lossy network conditions, they may perform worse when used in near lossless network conditions. However, if a thin-client approach does indeed have inferior performance compared to using a native browser, users will be understandably reluctant to adopt a thin-client computing model despite its other benefits.

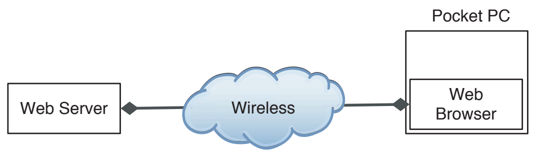

Figure 1: Traditional Fat Client

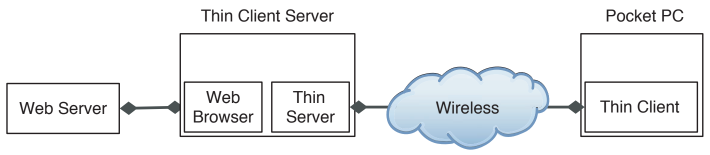

Figure 2: Thin Client

We explore the performance of thin clients in both simulated and real Wi-Fi network environments and quantitatively demonstrate for the first time that thin-client approaches can provide superior web browsing performance even in lossless Wi-Fi network environments. For our study, we focus on web browsing performance because of the importance and ubiquity of web applications. Previous studies show that web traffic constitutes the overwhelming majority of traffic in wireless networks [3, 15, 34]. We compare popular thin-client approaches embodied by Citrix MetaFrame and Microsoft Terminal Services, which represent the dominant commercial thin-client products in the marketplace. Because these thin-client systems are proprietary and closed-source, we obtained our results using a novel application of slow-motion benchmarking [25]. This provides a non-invasive measure of thin-client performance using network monitoring in a way that properly accounts for client processing time, which may be significant in the case of PDAs. We contrast thin-client performance with traditional fat client approaches in combination with a number of different web browsers, including Microsoft Internet Explorer, Mozilla, and NetFront. Our results show that thin clients perform better than fat clients even when they send more data during web browsing. Furthermore, they provide better web browsing functionality, correctly displaying web content on a PDAs that is otherwise not viewable using locally running native web browsers on PDAs. We analyze the differences in the underlying mechanisms used by various thin-client platforms and explain the fundamental characteristics of these approaches that surprisingly result in superior 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 Wi-Fi wireless PDAs using thin-client systems versus fat clients running native web browsers. For our thin-client systems, we compared the performance of Citrix MetaFrame XP and Microsoft Windows 2000 Terminal Services, the two most popular commercial thin-client products in the marketplace. In this paper, we also refer to the thin-client systems by their remote display protocols, which are Citrix ICA (Independent Computing Architecture) and Microsoft RDP (Remote Desktop Protocol), respectively. To evaluate their performance, we designed an experimental Wi-Fi (IEEE 802.11b) testbed and various experiments to assess both thin-client and native web browsing performance. 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, which includes both simulated and real Wi-Fi environments. Section 2.3 discusses the mix of benchmarks used in our experiments. We focused on the performance of widely deployed commercial solutions in all of our experiments to provide representative and realistic results.

2.1 Measurement Methodology

Because thin-client systems are designed and used very differently from traditional desktop systems, quantifying and measuring their performance effectively can be difficult. In traditional 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 can be completely decoupled from the display subsystem; therefore, the display updates visible on the client side may be asynchronous with the application events. Furthermore, display updates may be merged or even discarded in some systems to conserve network bandwidth.

To benchmark a conventional system, one can simply script a sequence of application events and let the system being tested execute the script as fast as it can. The result can generally be correlated with the user experience. However, in thin-client systems, because the application processing on the server can be out of sync with the display subsystem on the client, standard application benchmarks effectively measure only the server's application logic performance and do not accurately reflect user perceived visual performance at the client. The problem is exacerbated by the fact that many thin-client systems, including those from Citrix and Microsoft, are proprietary and closed-source, making it difficult to instrument them to obtain accurate, repeatable performance results.

To address these problems, we evaluate thin-client performance using slow-motion benchmarking, a non-invasive measurement technique previously developed by one of the authors [25]. Unlike previous slow-motion benchmarking methods, we apply this technique to web browsing performance in a novel way to also account for client processing time, which may be significant in the case of PDAs. We first describe how we use slow-motion benchmarking to measure web browsing performance, then discuss in Section 2.3 in further detail how specific web benchmarks were developed using this technique for our experiments.

To measure web browsing performance, we start with the conventional method of scripting a sequence of web pages using JavaScript to download one after another such that each subsequent page request is made as soon as the browser reports the current page is loaded [40]. We can then measure the total time required to download the web page sequence as a measure of the web browsing performance of the system. To account for the possibility that some display updates may not be fully displayed on the client even though the application processing completes, we also use a slow-motion version of the same benchmark. We create this version by altering the original benchmark to introduce delays between the separate visual components of the benchmark so that the display update for each component is fully completed on the client before the server begins processing the next display update. For example, delays are inserted between web pages for a web benchmark to ensure that each web page is completely displayed. We can then compare the measurements of the original benchmark with its slow-motion counterpart to determine whether any display updates were not fully displayed in the original benchmark, thereby providing a measure of the user-perceived performance based on the visual quality of display updates.

We measure benchmark performance in a non-invasive way by monitoring network activity. Since we could not directly peer into the black-box thin-client systems, our primary measurement technique was to use a packet monitor to capture resulting network traffic between the client and server. We used this technique to measure the latency incurred for running each benchmark as well as the data transferred between server and client. Given a benchmark that downloads a sequence of web pages, we measure the latency incurred for running the entire benchmark by monitoring network traffic to determine the time between the first and last packet sent during the benchmark. We also measure the total data transferred between server and client during the benchmark. We then run the slow-motion version of the benchmark and measure the data transferred between server and client during the slow-motion benchmark. For the slow-motion benchmark, 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. To derive a measure of performance that accounts for any missing display updates in execution of the original benchmark, we scale the latency of the original benchmark by the ratio of the data transferred during the slow-motion benchmark and the data transferred during the original benchmark. This latency is computed as follows:

$$\text{Latency} = \text{Latency}_{\text{original}} \times \frac{\text{DataTransferred}_{\text{slowmotion}}}{\text{DataTransferred}_{\text{original}}} \tag{1}$$

Our measurement technique makes four important assumptions. First, our approach assumes that if all display updates are completely processed and displayed, the amount of data transferred should remain about the same regardless of the rate at which the benchmark is run. Our previous work [25, 39] demonstrates that this is in general a valid assumption for existing thin-client systems such as Citrix MetaFrame and Microsoft Terminal Services, which do not employ highly adaptive compression techniques that might result in different degrees of compression depending on the rate at which display updates are generated.

Second, our approach assumes that display updates that have been sent from server to client are completely processed by the client and displayed. Furthermore, we assume serialization of display updates such that display updates sent to the client earlier are processed by the client before display updates sent later. If display updates sent to the client are discarded by the client after they are sent, the user perceived visual quality of a benchmark could be affected without any indication in the measurement of data transferred that some display updates were discarded. In addition, if the client could discard display updates after they have been sent, the client could then be ready to process the next set of display updates without incurring any client processing time for the discarded display updates, making the thin-client system appear faster in downloading a sequence of web pages without any indication that display quality was being sacrificed.

Our discussions with engineers of proprietary thin-client systems such as those from Citrix and Microsoft indicate that these systems do not discard display updates at the client after they have been sent. As a result, measuring the data transferred does provide an indicator of the user perceived visual quality while running a web benchmark. Furthermore, it is not possible for the client in a thin-client system to jump ahead and process newer display updates without having completed the older display updates. This ensures that our technique of measuring the latency of processing a sequence of web pages on a thin client implicitly accounts for client processing time as well, since earlier display updates need to be processed before subsequent ones. Note that we are only referring here to the ability of the client in a thin-client system to discard display updates. The server in a thin-client system may discard display updates before they are sent to the client when web pages are processed in rapid succession, but those discards would be reflected in a difference in the data transferred for the benchmark and its slow-motion version.

Third, our approach assumes that any client processing time that may be required after the last data packet is sent during the benchmark is relatively small and negligible. Since we are monitoring network traffic to measure latency, any client processing that occurs after the last packet is sent in rendering the last display update of the benchmark is not accounted for in our measurements. Only the client processing for the last display update on the last web page is unaccounted for; client processing time for any intermediate web pages is fully accounted for using our measurement technique. This processing time after the last display update for the benchmark can be made negligible compared to the overall benchmark latency by simply using a large enough number of web pages in the benchmark.

Fourth, our approach assumes that any reduction in latency due to display updates not being sent while running the original benchmark can be accounted for by linearly scaling the latency proportionally to any additional data transferred while running the slow-motion benchmark case. Since this is a latency estimation technique, this accounting is generally more accurate when the difference in the data transferred between the original and slow-motion versions of the benchmark is not large. Our measurements in Section 3 show that this was typically the case for our experiments.

More generally, the latency in running a benchmark using a thin client increases with the number of display updates that need to be processed and also increases with the size of the display update that needs to be processed for the same type of display update. However, different display updates may require different amounts of processing time per pixel. For example, some thin clients use display protocols that provide primitives for directly encoding text so display updates with text send such information as encoded text primitives rather than low-level raw pixels. On the other hand, thin clients typically do not provide special primitives for drawing images and typically decode and process images on the server and send the corresponding raw pixels to the client for display. As a result, for a thin client, text display updates typically require more client processing time per pixel than image display updates, which are already decoded and just need to be displayed.

In the case of web browsing a set of web pages, web pages typically consist of a container HTML page that is primarily text and a set of embedded objects that are most often images. If a web page in a scripted sequence of web pages is not fully displayed on a thin client, it is due to the fact that the next web page starts processing before the last display updates for the current web page have been fully displayed. This means that the missing display updates correspond to embedded images on the web page, which have a lower per pixel client processing cost than the HTML container page. By scaling latency linearly based on the difference in data between running the original and slow-motion version of a web benchmark, we provide a slower more conservative estimate of thin-client performance.

Note that this line of reasoning does not apply to measuring the performance of a fat client running a native web browser. With a fat client, decoding of embedded GIF and JPEG images occurs on the client as opposed to on the server. Such decoding may be more expensive per pixel than processing an HTML container page. As a result, applying our slow-motion linear scaling technique to a fat client running a native web browser may provide a faster estimate of the latency of such a system if the benchmark executed does not fully display the web pages on the client.

An alternative approach one might consider in non-invasively measuring web browsing performance on a thin client would be to measure the latency between the first packet and last packet transferred between server and client for each web page in the slow-motion version of the benchmark, then summed up the latencies to determine the overall latency of the benchmark. However, this measurement does not account for the client processing time on each page that occurs after the last packet of a screen update is sent to the client until the update is actually completely drawn to the client screen. If servers and clients have similar processing capabilities, it may be possible to neglect this client processing time in measuring thin-client performance. However, because client processing time can be significant in the case of PDAs, this approach would not work well for PDAs and would underestimate the web browsing latency of thin-client systems.

Our novel application of slow-motion benchmarking allows us to measure thin-client performance without any invasive modification of thin-client systems. As a result, we are able obtain our results without imposing any additional performance overhead on the systems measured. More importantly, the techniques make it possible for us to measure popular but proprietary thin-client systems, such as those from Citrix and Microsoft.

2.2 Experimental Testbed

Figures 3 and 4 show the simulated and real Wi-Fi isolated experimental testbeds we used for our experiments. Each network testbed consisted of a client machine, a packet monitor, a thin-client server, and a web server. The simulated Wi-Fi network testbed used a desktop PC as the client and a network emulator machine to emulate a Wi-Fi network environment. Both machines were connected using 100 Mbps Ethernet. We used the simulated Wi-Fi network to enable more flexible experimentation with different network characteristics during our study. The real Wi-Fi network testbed used a Pocket PC handheld PDA as the client and a Lucent Orinoco AP-2000 wireless access point to provide the Wi-Fi network environment. The client connected to the access point using a Dell Compact Flash 802.11b wireless ethernet card. The packet monitor, thin-client server, and web server were the same for both testbeds. We used two testbeds in part to allow experimentation with different client configurations.

The features of each system are summarized in Table 1. Except for the client machines, all machines are IBM Netfinity PCs, each with dual 933 MHz Pentium III CPUs, 512 MB RAM, 9 GB disk, and 10/100BaseT NICs. The desktop PC client is a Micron Client Pro PC with a 450 MHz Pentium II CPU, 128 MB RAM, and 14.6 GB disk. Although the desktop PC is quite modest by current desktop PC performance standards, the hardware configuration was selected to provide a more even comparison with a modern PDA. The Pocket PC PDA is a Dell Axim X5 with a 400 MHz Intel XScale PXA255 CPU and 64 MB RAM. Because all tests with the wireless network were conducted within ten feet of the access point, we considered the amount of packet loss to be negligible in our experiments. To provide similar network conditions simulated Wi-Fi testbed, we used the network emulator to limit available bandwidth to a maximum of 6 Mbps. 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 [1].

For simplicity and good network performance, both network testbeds were configured using 100BaseT full duplex switched network connections between all wired testbed machines. In this configuration, network traffic was routed through the packet monitor to accurately capture network data. Furthermore, when measuring the performance of fat clients, we reconfigured our thin-client server to simply act as a packet forwarding machine to provide access to the web server directly from the client machine. The added latency of routing through the machines was measured and is negligible.

Whenever possible, we used common system configuration options, common applications, and common thin-client configuration options. When it was not possible to configure all the platforms in the same way, we generally used default settings. Apache 1.3.27 was used as the web server for all of the web benchmarks. All of the fat-client and thin-client systems were run in Microsoft operating system environments. All of the systems were configured with 1024x768 display resolution. Since the Pocket PC PDA only has 240x320 screen resolution, scrolling around the display was necessary to see all of the content displayed.

All of the thin-client systems were configured with 1024x768 display resolution and default settings of compression enabled, memory caching enabled, and disk caching disabled. Data encryption was disabled to provide a fair comparison with fat clients using insecure unencrypted HTTP connections. Because the Pocket PC PDA does not support more than 16-bit color depth, Citrix ICA was configured with 16-bit color depth on all platforms. Microsoft RDP was configured with 8-bit color depth since that was the maximum color depth supported by the version of RDP used with Windows 2000 Terminal Services, which was used for the thin-client server. All web and thin-client caches were cleared before each experiment.

To account for performance differences due to different web browsers, we experimented with two different web browsers for each platform. For the Pocket PC, we used both Microsoft Pocket PC 2003 Internet Explorer, the latest version Pocket PC browser from Microsoft, and ACCESS' popular NetFront 3.0 web browser. For the desktop PC and thin-client systems, we used both Microsoft Internet Explorer 6 and Mozilla 1.4. The use of Internet Explorer on both platforms provides a common basis for performance comparison. Mozilla was also used with the desktop PC since it represents the next most widely used web browser. However, since no version of Mozilla was available for Pocket PC, we used another popular Pocket PC web browser, NetFront, to compare the performance for different browsers. All of the web browsers used were configured with full screen 1024x768 browser window sizes with default cache settings enabled in each browser. Persistent HTTP 1.1 was used for all experiments for best performance.

2.3 Application Benchmarks

To measure web performance of both thin-client and traditional fat-client systems, we used three web browsing application benchmarks, representative of general consumer web content, clinical image content as would be viewed by medical professionals, and

Role / Model	Hardware	Operating System	Software
PC Thin Client Micron Client Pro	450 MHz Intel PII 128 MB RAM 10/100BaseT NIC	MS Win XP Professional	Citrix ICA Win32 Client MS RDP5 Client MS Internet Explorer 6 Mozilla 1.4
Pocket PC Client Dell Axim X5	400 MHz Intel XScale 64 MB RAM Dell 802.11b Wireless CF Card	MS Pocket PC 2003	Citrix ICA Client MS RDP5 Client MS Pocket PC 2003 Internet Explorer Access NetFront 3.0
Packet Monitor IBM Netfinity 4500R	Dual 933 MHz Intel PIII 512 MB RAM 10/100BaseT NIC	Debian Linux Testing (2.4.20 kernel)	Ethereal Network Analyzer 0.9.13
Benchmark Server IBM Netfinity 4500R	Dual 933 MHz Intel PIII 512 MB RAM 10/100BaseT NIC	Debian Linux Testing (2.4.20 kernel)	Apache Web Server 1.3.27
Thin Client Server/ Packet Forwarder IBM Netfinity 4500R	Dual 933 MHz Intel PIII 512 MB RAM 10/100BaseT NIC	MS Win 2000 Server Debian Linux Unstable (2.4.20 kernel)	Citrix MetaFrame XPe MS Win 2000 Terminal Services MS Internet Explorer 6 Mozilla 1.4
Network Emulator IBM Netfinity 4500R	Dual 933 MHz Intel PIII 512 MB RAM 10/100BaseT NIC	MS Win 2000 Server Debian Linux Unstable (2.4.20 kernel)	NISTNet 2.0.12

Table 1: Testbed Machine Configurations

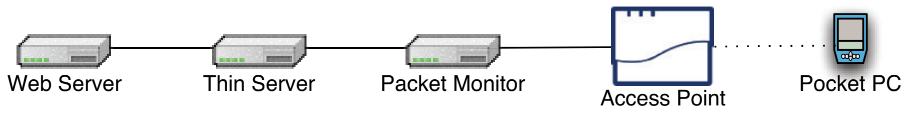

Figure 3: Real Wi-Fi Network Testbed

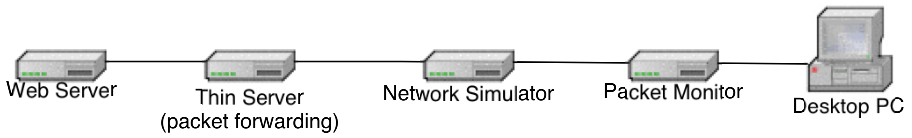

Figure 4: Simulated Wi-Fi Network Testbed

a clinical information system as viewed by medical professionals. We refer to the benchmarks as i-Bench, RSNA, and WebCIS, respectively. We describe each benchmark and how they were modified to create respective slow-motion versions of each benchmark.

The i-Bench general consumer web content benchmark we used is based on the Web Text Page Load test from the Ziff-Davis i-Bench 1.5 benchmark suite [40]. It consists of a JavaScript controlled load of 54 web pages from the web benchmark server. The pages contain both text and bitmap graphics, with pages varying in the proportions of text and graphics. The graphics are embedded images in GIF and JPEG formats. We modified the original i-Bench benchmark for slow-motion benchmarking by introducing delays of several seconds between pages using 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 the benchmark, and then used the timestamps of the first and last packet associated with the benchmark to obtain latency measures for the benchmark, which are then scaled based on Equation 1.

The RSNA clinical web content benchmark we used is a multi-page test that downloads a sequence of 20 primarily graphical web pages demonstrating real-time contrast enhancement of mammographic features via multiscale analysis. This benchmark was created from pages used at an exhibition at the Radiological Society of North America (RSNA) Scientific Assembly and Annual Meeting [17]. Each page contains 2 mammographic images. The first mammogram image is a static bitmap. The second is generated in real-time by the web server through a multiscale wavelet enhancement of the first mammogram image. The image enhancement calculations are performed by a common gateway interface (CGI) script running on the web server. The CGI computes the image enhancement and displays the results on the web page. These images are representative of the kinds of image processing activities that are anticipated to become commonplace in the clinical practice of radiological diagnosis in the near future. Similar to the i-Bench benchmark, we modified the original RSNA 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 the benchmark, and then used the timestamps of the first and last packet associated with the benchmark to obtain latency measures for the benchmark, which are then scaled based on Equation 1.

The WebCIS clinical information system (CIS) benchmark is a multi-page test that downloads a sequence of 18 web pages from New York Presbyterian Hospital's Web-based clinical information system (WebCIS) [6, 10]. WebCIS displays data from a variety of clinical data sources including results from laboratory, radiology, and cardiology departments. These pages contain primarily textual data in free text and in tabular form. Navigation and forms submission is JavaScript driven. The pages selected for the bench-

Name	Description
PC FAT IE	PC running native Internet Explorer
PC FAT MOZ	PC running native Mozilla
PC ICA IE	PC running Citrix ICA client w/ Internet Explorer
PC ICA MOZ	PC running Citrix ICA client w/ Mozilla
PC RDP IE	PC running Microsoft RDP client w/ Internet Explorer
PC RDP MOZ	PC running Microsoft RDP client w/ Mozilla
PDA FAT IE	PDA running native Internet Explorer
PDA FAT NF	PDA running native NetFront
PDA ICA IE	PDA running Citrix ICA client w/ Internet Explorer
PDA ICA MOZ	PDA running Citrix ICA client w/ Mozilla
PDA RDP IE	PDA running Microsoft RDP client w/ Internet Explorer
PDA RDP MOZ	PDA running Microsoft RDP client w/ Mozilla

Table 2: Platform Configurations Used

mark were based upon common page sequences derived through CIS log analysis [5]. CIS log analysis is a method based on data mining and Web usage mining used to discover patterns of CIS usage. One of the techniques used in this method, sequential pattern discovery, was applied to a year's worth of WebCIS logs to identify common sequences of data access in WebCIS. We used these typical sequences in the WebCIS benchmark in order to reflect the usage patterns of actual clinical users of WebCIS. Similar to the previously discussed benchmarks, we modified the original WebCIS benchmark for slow-motion benchmarking by introducing delays of several seconds between pages 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 the benchmark, and then used the timestamps of the first and last packet associated with the benchmark to obtain latency measures for the benchmark, which are then scaled based on Equation 1.

3. MEASUREMENTS

We ran the three web benchmarks on both fat-client and thin-client systems running on both the desktop PC and handheld PDA with the three different web browsers and measured their resulting performance. We report results for the twelve different combinations shown in Table 2. The primary performance measurements for running each web benchmark are presented in terms of average web page download latencies for each system. For completeness, we present data showing three measurements for each benchmark: (1) the average web page download latency computed using Equation 1, (2) the average amount of data transferred per web page while running the slow-motion version of the benchmark, and (3) the ratio of the data transferred when running the slow-motion version of the benchmark versus the original benchmark, which we refer to as the *latency scale factor*. These measurements provide the first quantitative performance comparisons of handheld thin-client systems in Wi-Fi network environments. They also provide some useful data about the performance of different web browsers in different system configurations.

Figures 5 to 7 show the measurements for running the i-Bench web benchmark on each of the twelve platform configurations. Figure 5 shows the average web page latency for running the i-Bench benchmark on each platform. 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 six seconds [14]. All of the platforms provide average web page latencies of less than six seconds and all of the PC platforms provide average web page latencies of less than one second. However, on the PDA, only the thin-client platforms provide average web page latencies of less than one second. On the PDA, ICA MOZ, RDP IE, and RDP MOZ all provide average web page latencies of a second or less.

More importantly, on both the PC and the PDA, Figure 5 shows that the thin-client systems provide lower web browsing latencies than the fat-client systems when using the same browser. On the PC, ICA IE and RDP IE are 20 to 40 percent faster than FAT IE while ICA MOZ and RDP MOZ are roughly three times faster than FAT MOZ. The performance difference between the thin-client and fat-client approaches is even more substantial on the PDA. On the PDA, ICA IE is almost three times faster than FAT IE while RDP IE is more than seven times faster than FAT IE. While the Mozilla-based systems are slower than their Internet Explorer counterparts on the PC, even the Mozilla-based thin-client configurations significantly outperform FAT IE on the PDA. On the PDA, while FAT NF provides better fat-client performance than FAT IE, FAT NF is still much slower than all of the thin clients, with the Mozilla-based thin clients still more than twice as fast as FAT NF.

Figure 5 also shows some performance inversions between the PC and PDA platforms. On the PC, Internet Explorer performs better than Mozilla on all platforms. On the PDA, the non-Microsoft NetFront browser used in FAT NF outperforms the Internet Explorer-based FAT IE. On the PC, ICA performs better than RDP but on the PDA, RDP performs better than ICA.

Figure 6 shows the data transferred for running the slow-motion version of the i-Bench benchmark. For each platform, the data transferred is generally similar for both the PC and the PDA. As would be expected, all of the fat clients running native web browsers transfer roughly the same amount of data on both the PC and PDA. ICA IE on PC and PDA also send roughly the same amount of data, and ICA MOZ on PC and PDA also send roughly the same amount of data. However, RDP transfers very different amounts of data on PC versus PDA, with the data transferred when running on the PDA being much less than running on the PC. RDP IE on PDA transfers almost three times less data than RDP IE on PC.

Figure 6 shows that there are also large differences in the amount of data transferred across different platforms. On the PC, all of the fat clients transfer less data than all of the thin clients, with ICA transferring the most amount of data, almost twice as much data as the fat clients. However, on the PDA, RDP sends the least amount of data among all of the platforms. RDP IE transfers less than half the amount of data as the fat clients. ICA still sends the most amount of data on the PDA, sending almost twice as much data as the fat clients. Among the thin clients, there are also differences in the amount of data transferred depending on the web browser used. For a given thin client, using Internet Explorer resulted in less data transferred than using Mozilla. For instance, ICA MOZ transfers roughly ten percent more data than ICA IE on both PC and PDA.

Figures 5 and 6 taken together show that there is generally little correlation between the latency and the amount of data transferred for each platform. The fat clients had the worst latency performance for both PC and PDA yet generally sent less data, sending the least amount of data for the PC. ICA transferred the most

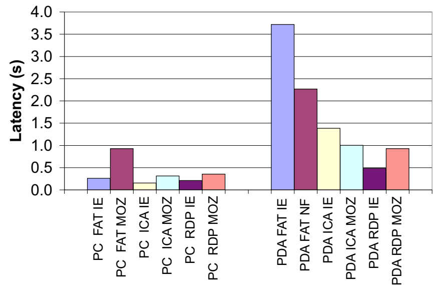

Figure 5: i-Bench Page Latency

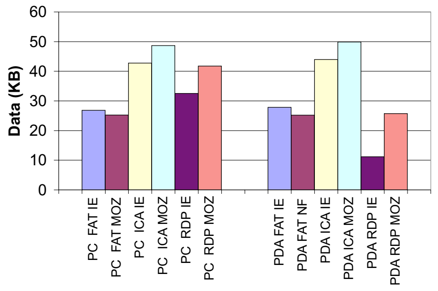

Figure 6: i-Bench Page Data Transfer

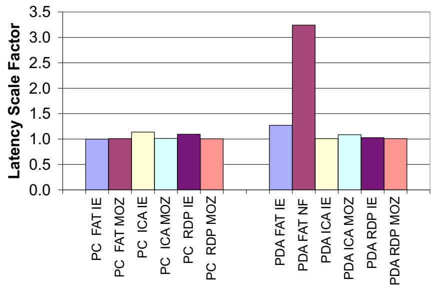

Figure 7: i-Bench Latency Scale Factor

amount of data but had better performance than the fat clients. Only in the case of RDP on the PDA did a platform both have the small latencies and the least amount of data transferred. However, RDP still outperformed the fat clients on the PC even though it transferred more data in that case.

Figure 7 shows the latency scale factor for each platform, which is the ratio of data transferred when running the original i-Bench benchmark versus running the slow-motion version of i-Bench. As one would expect, all of the latency scale factors are at least one, indicating that the slow-motion version of the benchmark always transferred at least as much data as the original version of the benchmark. For all of the thin-client platforms on both the PC and PDA, the latency scale factors were all close to one. The largest scale factor across all platforms was less than 1.15, which means that the amount of display update data not sent when running the original benchmark versus the slow-motion version of the benchmark was less than fifteen percent. This indicates that despite the decoupling between application processing and client display for the thin clients, most of the display updates were completely sent and processed by the thin client even without the delays used in the slow-motion version of the benchmark. Surprisingly, the largest latency scale factors occurred for the fat clients running on the PDA. FAT IE on the PDA had a modest latency scale factor of 1.27 while FAT NF on the PDA had a latency scale factor of more than three. A substantial amount of the FAT NF display updates are not properly sent during the i-Bench benchmark.

Note that even if we did not use the latency scale factor for FAT IE on the PDA, its average web page latency would still be substantially worse than the latency for the thin-client systems. If the latency scale factor for FAT NF was not used, its average web page latency would no longer be worse than the thin-client systems, but that would not be an accurate comparison. As a more conservative measure, we also ran the slow-motion i-Bench on FAT NF and measured the latency between the first packet and last packet transferred between server and client for each web page, then summed up the latencies to determine the overall latency of the benchmark. Although this measure does not account for client processing time, the resulting FAT NF average web page latency computed using this method was still no better than the latency of ICA IE on PDA using the latency scale factor. This shows that FAT NF is still slower than the thin-client approaches even when we unfairly ignore substantial portions of its client processing time.

Figures 8 to 10 show the measurements for running the RSNA web benchmark on each of the twelve platform configurations. Figure 8 shows the average web page latency for running the RSNA benchmark on each platform. All of the PC platforms and all of the thin-client systems on the PDA provide average web page latencies well under six seconds. However, none of the platforms provided average web page latencies less than one second. On the PDA, FAT IE and FAT MOZ provided web page latencies over eight seconds.

More importantly, on both the PC and the PDA, Figure 8 shows that the thin-client systems overall provide lower web browsing latencies than the fat-client systems when using the same browser. On the PC, RDP IE had the lowest latencies of all of the platforms and RDP MOZ had the lowest latencies of all of the Mozilla-based platforms. However, while there are some differences in latencies among different platforms on the PC, these differences are relatively small. On the PDA, the latency differences among different platforms was much more substantial. The PDA fat clients were generally more than two times slower than the thin clients and in the worst case, more than five times slower than the fastest thin client. On the PDA, RDP IE provided the fastest performance. However, RDP MOZ was roughly fifty percent slower than the ICA-based platforms. Based on the industry quality goal of download times of less than six seconds, only the thin clients provided acceptable web browsing performance on PDAs.

Figure 8 also shows some performance inversions between the PC and PDA platforms. On the PC, Internet Explorer performs better than Mozilla on all platforms. On the PDA, the non-Microsoft NetFront browser used in FAT NF outperforms the Internet Explorer-based FAT IE. On the PC, ICA performs worse than RDP but on the PDA, RDP has higher variance in performance when used with different web browsers and does noticeably worse than ICA when using Mozilla.

Figure 9 shows the data transferred for running the slow-motion version of the RSNA benchmark. For each platform, there is some variability in the amount of data transferred using PC versus PDA. Figure 9 shows that ICA IE transfer approximately the same amount of data as the fat-client systems. However, ICA MOZ transfers significantly less data. RDP on both the PC and the PDA also transfer less data than their fat client counterparts.

This difference in data transfer between fat and thin clients may seem surprising. However, in the RSNA benchmark, the images transferred to the fat clients were sent as uncompressed GIFs. When these same images were sent to the thin clients through the remote display protocol, these images were automatically losslessly compressed. One might assume that this difference in data transfer would account for the performance difference of the fat and thin clients.

However, Figures 8 and 9 taken together show that there is generally little correlation between the latency and the amount of data transferred for each platform. For the PC, all of the platforms had similar latencies but there is a factor of two difference in the amount of data transferred between FAT MOZ, which sent the most data, and RDP IE, which sent the least data The fat clients had slightly better performance than ICA on the PC but sent more data. Only in the case of RDP IE on the PDA and the PC did a platform both have the small latencies and the least amount of data transferred. However, RDP MOZ sent the second least amount of data on the PDA and performed worse than both ICA IE and ICA MOZ.

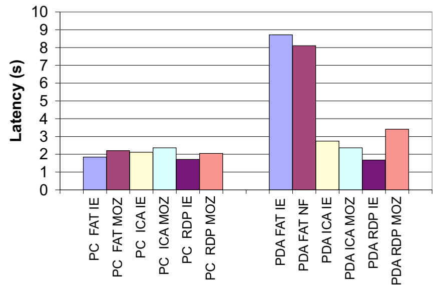

Figure 8: RSNA Page Latency

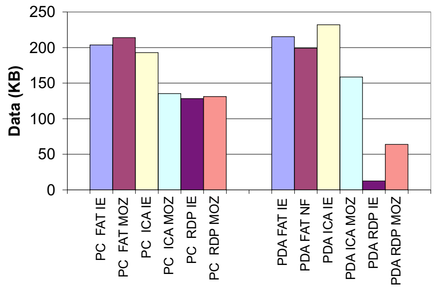

Figure 9: RSNA Page Data Transfer

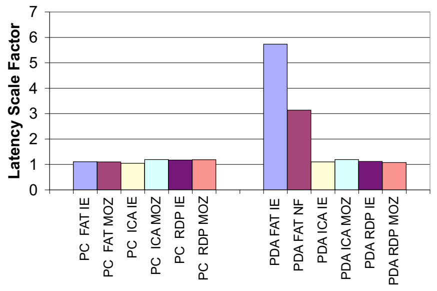

Figure 10: RSNA Latency Scale Factor

Figure 10 shows the latency scale factor for each platform, which is the ratio of data transferred when running the original RSNA benchmark versus running the slow-motion version of RSNA. As one would expect, all of the latency scale factors are at least one, indicating that the slow-motion version of the benchmark always transferred at least as much data as the original version of the benchmark. However, for all of the thin-client platforms on both the PC and PDA, the latency scale factors were all close to one. The largest scale factor across all platforms was less than 1.2, which means that the amount of display update data not sent when running the original benchmark versus the slow-motion version of the benchmark was less than twenty percent. This indicates that despite the decoupling between application processing and client display for the thin clients, most of the display updates were completely sent and processed by the thin client even without the delays used in the slow-motion version of the benchmark. Surprisingly, the largest latency scale factors occurred for the fat clients running on the PDA. FAT IE and FAT NF on the PDA had latency scale factors of more than three, indicating that a substantial amount of their respective display updates are not properly sent during the RSNA benchmark.

As discussed in Section 2.1, our latency measurements are more accurate if the latency scale factors are not that large. Because of the large latency scale factors for FAT IE and FAT NF on the PDA, we also obtained a more conservative measure of their performance. We ran the slow-motion i-Bench on these platforms and measured the latency between the first packet and last packet transferred between server and client for each web page, then summed up the latencies to determine the overall latency of the benchmark. This measure does not account for client processing time. The resulting FAT IE and FAT NF average web page latencies computed using this method were still worse than the thin-client latencies shown in Figure 9. This shows that the fat clients are still slower than the thin-client approaches even when we unfairly compare the two by including thin-client client processing time and excluding substantial portions of fat-client client processing time.

Figures 11 to 13 show the measurements for running the WebCIS benchmark on each of the twelve platform configurations. Figure 11 shows the average web page latency for running the WebCIS benchmark on each platform. All of the platforms except for FAT IE and FAT NF on the PDA provided average web page latencies well under one second. On the PDA, data for FAT IE and FAT NF are not shown because the browsers were unable to complete the benchmark. For this benchmark, the platforms that completed the benchmark all provided acceptable web browsing performance based on the one-second download metric for providing an uninterrupted browsing experience.

While most of the platforms provided acceptable web browsing performance, Figure 11 shows that thin-client systems generally provide similar if not lower web browsing latencies than the fat-client systems when using the same browser. The only case in which the fat client was faster than the thin client was on the PC using Internet Explorer. On the PC, FAT IE was slightly faster than ICA IE and RDP IE. For all other cases, the thin clients performed better. Using Mozilla with the PC, RDP MOZ was the fastest and FAT MOZ was the slowest of all systems. On the PDA, RDP MOZ provided the fastest performance and had the smallest latencies across both PC and PDA systems. PDA ICA IE provided the worst performance among the thin clients, roughly twice as slow as RDP MOZ. However, even the slowest PDA thin client performed better than any of the PDA fat client approaches, since none of those systems could even complete the benchmark.

Figure 11 also shows some performance inversions between the PC and PDA platforms. On the PC, Internet Explorer performs better than Mozilla on all platforms. On the PDA, the thin clients using Mozilla performed better than those using Internet Explorer. In particular, the Microsoft RDP thin client performed better than any other system on the PDA using Mozilla, not Microsoft's own Internet Explorer. Similarly on the PC, ICA performed better than RDP using Internet Explorer, even though RDP and Internet Explorer are both from Microsoft. In contrast, RDP performed better than ICA using Mozilla.

Figure 12 shows the data transferred for running the slow-motion version of the WebCIS benchmark. For each platform, the data transferred is generally similar for the PC and PDA. As would be expected, all of the fat clients running native web browsers transfer roughly the same amount of data on both the PC and PDA. ICA IE on PC and PDA also send roughly the same amount of data, and ICA MOZ on PC and PDA also send roughly the same amount of data. RDP transfers slightly different amounts of data on PC versus PDA, with the data transferred when running on the PDA being much less than running on the PC. Figure 12 shows that there are also large differences in the amount of data transferred across different platforms. On the PC, all of the fat clients transfer approximately three times as much data as the thin clients. On the PC and PDA, ICA transfers the least amount of data among all of the platforms. RDP transfers slightly more data than ICA for both web browsers on both PC and PDA.

This large difference in data transfer between fat and thin clients is a particularly surprising result with WebCIS. Because WebCIS is a text rich application, one would likely come to the conclusion that a fat client should transfer less data than the graphical representation used by thin clients to represent the web page. However, like many modern web applications, WebCIS makes extensive use of JavaScript and other code that is executed in the browser. This code accounts for the surprisingly large amount of the data transferred to the fat clients. With thin clients, this code does not need to be transferred to the client and only the end results of the execution of the JavaScript need to be transmitted to the client. As a result, thin clients can transfer less data than fat clients even when primarily textual data is being displayed.

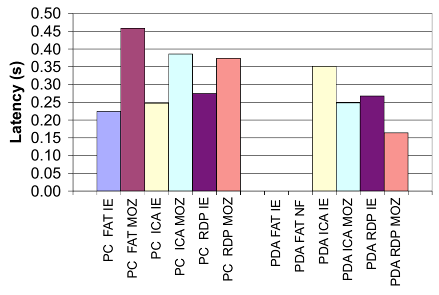

Figure 11: WebCIS Page Latency

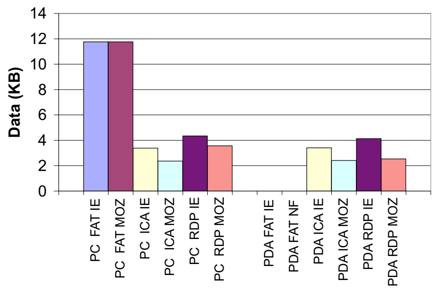

Figure 12: WebCIS Page Data Transfer

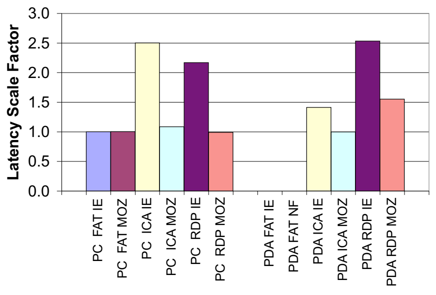

Figure 13: WebCIS Latency Scale Factor

Figures 11 and 12 taken together show that there is generally little correlation between the latency and the amount of data transferred for each platform. The fat clients varied widely in performance, yet transferred similar amounts of data. On the PC, FAT IE and FAT MOZ transfer the same amount of data yet FAT IE is more than twice as fast as FAT MOZ. On the PC, RDP IE is faster than RDP MOZ although RDP IE sends more data. On the other hand on the PDA, RDP IE is slower than RDP MOZ and RDP IE still sends more data. The results with the WebCIS benchmark are consistent with those for the i-Bench and RSNA benchmarks in that there is not much correlation in any of the results between the latency and the amount of data transferred for each platform.

Figure 13 shows the latency scale factor for each platform, which is the ratio of data transferred when running the original WebCIS benchmark versus running the slow-motion version of the WebCIS benchmark. As expected, all latency scale factors are at least one, indicating that the slow-motion version of the benchmark transfer at least as much data as the original version of the benchmark. The fat clients that completed the benchmark all had latency scale factors of one. Only the thin clients had larger latency scale factors, with the largest being 2.5. This indicates that there is some decoupling between the application processing and client display for the thin clients. The effect of this appears to be more significant in the WebCIS benchmark than in the previous benchmarks. The effect is also more significant when using thin clients with Internet Explorer instead of Mozilla to run the WebCIS benchmark.

4. INTERPRETATION OF RESULTS

Our measurements show that thin-client systems can provide functionally better web browsing and faster web page download latencies than fat-client systems, especially in the case of PDAs. These results are counterintuitive given that thin clients add an extra layer of software between client and web server, which it would seem should add extra latency to processing web pages. These results are also counterintuitive given that our measurements show that these thin-client systems provide faster web page download latencies even when transferring more data than their fat-client counterparts. To explain the reasons for the behavior shown in our measurements, we discuss four reasons that account for the performance differences between these systems and point to the benefits of the thin-client approach: limitations of PDA web browsers, distribution of client and server processing, and display size web browsing costs.

4.1 Limitations of PDA Browsers

One important reason why our measurements show that thin clients can provide better performance on PDAs is that web browsers that run natively on PDAs simply do not work well. Measurements shown in Figures 5 and 8 not only show that thin clients provide better performance than fat clients on PDAs, but that native web browsing on a PDA gives much worse perform than native web browsing on a desktop PC.

The fat-client model is inherently harder to support from a software development perspective. First, the model requires that each platform needs to be able to run its own web browser, which means that browsers must be developed for each platform. Supporting multiple browser versions for different platforms is certainly harder than supporting just one. This is exacerbated by the fact that a web browser is a complicated piece of software to start with. Second, getting a complicated web browser to work effectively on a PDA is a doubly challenging problem. Not only must complex browser functionality be stripped down enough to run in a PDA environment, but it must also be optimized to run in a much more resource constrained environment as well. Furthermore, as earlier PDAs were even more resource constrained than the relatively powerful Pocket PC model we used for our experiments, PDA web browser developers are also faced with the challenge of having to somehow evolve the web browser from an even more resource constrained environment while at the same time optimizing it for best performance.

The problem of developing effective PDA web browsers was evident by the poor functionality of these browsers in our experiments. We discuss three examples from our study. A first example is shown in Figures 7 and 10, in which both the Pocket PC Internet Explorer and NetFront PDA web browsers consistently had large latency scale factors. This indicates that they were not completely drawing the web pages as they cycled through the sequence of web pages in each benchmark. This behavior occurs because both browsers do not properly support the JavaScript OnLoad event, which is commonly used to perform some function only after a given web page has been completely downloaded and displayed. Pocket PC Internet Explorer and NetFront did not properly handle this event, which was used in our benchmarks to ensure that a web browser does not process the next web page until it completes the current one.

A second more egregious example of poor PDA web browser functionality is shown in Figure 11, which demonstrates the inability of Pocket PC Internet Explorer and NetFront to work with the WebCIS benchmark. This problem was again due to poor support for JavaScript by these PDA browsers. The JavaScript used in this benchmark is the same code used in the WebCIS web-based clinical information system [6], which is widely deployed and used at New York Presbyterian Hospital. The lack of browser functionality in PDA browsers means that these devices could simply not be used to access a production web-based information system.

A third example is that the Pocket PC Internet Explorer browser advertises its ability to a web server that it can use HTTP 1.1 and persistent connections. However, the behavior it exhibits is completely nonstandard and is in fact more like that of using nonpersistent connections. In our experiments, for each request for a page of HTML or an image, Pocket PC Internet Explorer opened a new connection. After the data for the object is received, it closed the connection using a TCP reset, forcing the connection to close abnormally, resulting in a separate connection for each web object.

In contrast to the fat-client model, thin clients do not require the development and maintenance of complex software on multiple platforms. Only a simple thin-client application needs to run on each client platform. Since all application logic resides on the server, only a web browser that runs on the server is required, despite having a plurality of different client devices. Thin clients can then leverage substantial existing investments in PC web browser technology. These investments provide for a more optimized web browser with a more highly tuned rendering engine, which can be effectively used via a thin-client system on any client. This results in much better performance on PDAs than running native PDA web browsers that may not function well in the first place. More importantly, there are production web applications such as WebCIS with a significant investment in their development that do not run on web browsers designed for PDAs and would require significant modification in order to support those browsers. These applications are also often not tested on PDA browsers. In contrast, thin clients leverage desktop web browsers to work seamlessly with such production systems without any modifications.

4.2 Distribution of Client and Server Processing

Another important reason why our measurements show that thin clients can provide better performance than fat clients on both PCs and PDAs is how each approach distributes client and server processing. A fat client places all of the web browsing processing on the client. This places complex browser processing on a client, which is often slower than typical servers. In particular, PDAs are necessarily resource constrained devices given their size and power requirements. Servers on the other hand do not have these limitations and can be larger and more powerful machines.

In contrast, a thin client runs its web browser application logic on the more powerful server while only running a simple thin-client application for processing display updates on the client. The thin-client model provides a better distribution of web browsing processing requirements by putting the complex, more resource intensive processing on the server. Our measurements showed that the thin clients outperformed the fat clients when using the PC client. This performance difference was largely due to the fact that the web browser was running on a faster server when using the thin client. For example, for the i-Bench benchmark, Internet Explorer running on the server was twice as fast as the same browser running on the slower PC client. This difference in performance more than compensates for the extra processing involved with the extra layer of software introduced with the thin-client systems.

The difference in processing power between server and client was not simply an issue of CPU speed, but it was also an issue of CPU architecture functionality. The Pentium III CPU in the server was designed as a powerful server CPU. In contrast, the Dell Axim PDA used for our study is based on the Intel XScale PXA255 CPU. The Dell Axim PDA is considered one of the more high performance PDAs available today, but its XScale CPU was designed as a lower cost, low power, integrated CPU to work in the context of mobile devices. These are different design goals and result in different functionality. In particular, the Pentium CPU in the server provides MMX instructions, which can be used to provide very fast image processing operations. The XScale CPU does not provide this functionality. Internet Explorer takes advantage of MMX instructions when available to dramatically improve the speed of GIF and JPEG decoding and processing. As a result, GIF and JPEG decoding and processing on the server was well more than an order of magnitude faster than such processing on the PDA, even though the difference in CPU clock rate of the server CPU and PDA CPU was only a factor of two. The thin-client approach takes advantage of this speed difference since its web browser processing occurs on the server. In contrast, the fat-client approach is limited by the client and cannot take advantage of the special CPU instructions available on the server to optimize GIF and JPEG processing. Thin clients provide a model of distributing client and server processing and functionality that matches well with the underlying client and server hardware resources.

4.3 Display Size Web Browsing Costs

Another reason why our measurements show that thin clients can provide better performance than fat clients on PDAs is how each approaches display updates. When accessing a web page, a fat client sends all of the data related to that web page, regardless of whether or not the entire page is viewed. This aspect is particularly important when considering the limited display sizes on PDAs. Because screen sizes are so limited on PDAs, frequently a large portion of a web page is never viewed by the user, but is sent to the client web browser anyways. This limitation is fundamental to the model of HTTP.

In contrast, with thin clients, the model is based upon display updates. The server does not need to send to the client what is not being displayed. Because of the small display size of PDAs, this gives an opportunity to optimize what information is sent from server to client for each display update. In particular, a thin-client system can avoid sending display updates that are not actually viewed and only send data associated with display updates that are visible on the client. This server-side clipping optimization not only reduces the amount of data that needs to be sent, it also reduces the amount of display update processing required on a client. Both of these benefits can result in improved performance for thin clients.

As shown in Figures 5, 8, and 11 RDP provides the best performance on the PDAs for all three benchmarks, in part because it uses this display clipping optimization. The impact of the optimization can be seen by comparing the amount of data transferred using RDP on the PC versus the PDA in Figures 6, 9, and 12. For all of the benchmarks, RDP sends less data when used from the PDA versus the PC because of the PDA's much smaller display size. This effect is most pronounced with the RSNA benchmark, where PDA RDP IE transfers eight times less data than PC RDP IE.

Not all thin clients provide this display clipping optimization. Figures 6, 9, and 12 show that ICA transfers roughly the same amount of data on both the PC and the PDA. ICA sends the entire display to the client even though the viewable region is smaller than the display. ICA then pans around the desktop through clipping the viewable region on the client side. As a result, ICA sends more data than RDP since it performs the clipping after transmission to the client.

4.4 Connection Model

One final reason why thin clients can provide better performance than fat clients on wireless PDAs is due to their connection model. A web browser opens up one or more TCP connections to download a web page and its embedded objects. Even if persistent HTTP is used, at least one connection needs to be opened up for each domain name. A DNS lookup may be needed for each domain name as well. With a fat client running a native PDA browser, these DNS lookups and multiple connection setups must occur over the wireless network. If the wireless network is lossy, this behavior can severely degrade web browsing performance [38]. In contrast, a thin client approach runs the web browser on the server and only needs a single persistent connection to the thin-client server, which then opens up one or more connections over the wired network to any web servers. This eliminates the cost of DNS lookups and multiple connection setups over the wireless network. The cost of maintaining a connection once it has been setup is much lower than the cost of connection setup especially in the presence of packet loss due to the larger timeouts used by the TCP exponential backoff mechanism during connection setup.

In our experiments, DNS lookups and connection setup cost was not a significant factor in the performance difference between fat clients and thin clients. However, our experiments are not representative of more realistic web surfing behavior when multiple web servers are accessed, requiring multiple DNS lookups. Furthermore, our experiments were conducted under near ideal wireless network conditions where the client device was located right next to the Wi-Fi access point. Network conditions would not be as ideal in practice in the presence of interference from physical barriers and other networks. In those circumstances, the cost of DNS lookups and connection setup would be a more significant factor that would further increase the performance improvement of thin clients over fat clients.

5. RELATED WORK

In addition to the systems discussed in this paper, several other thin-client and remote display systems have been developed. These include Sun Ray [30, 32], Tarantella [28, 31], VNC [2, 27], X [29] and extensions such as low-bandwidth X (LBX) [4] and Kaplinsk's VNC tight encoding [13], as well as remote PC solutions such as Laplink [18] and PC Anywhere [33]. Several studies have examined the performance of thin-client systems [16, 30, 35, 22, 36, 23, 24, 25, 37, 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 performance of thin-clients in wireless networks or PDAs. More recently, another study co-authored by one of the authors of this paper demonstrated that thin clients can outperform fat clients in lossy wireless networks [38] due to several factors, including lower connection setup costs and the ability to ignore previous display updates that may have been lost. That study did not consider using PDAs and the resulting performance and functionality impact.

Other approaches to improving the performance of mobile wireless web browsing have focused on using transcoding and caching proxies in conjunction with the fat client model [19, 12]. Top Gun Wingman was a proxy-based system that pushed some of the application complexity to a back end proxy server [9]. The proxy transcoded images into scaled reduced fidelity images and translated HTML into a simplified markup language. The system effectively requires a web browser reimplementation by introducing a HTML parser in the proxy and a specialized application for display and layout at the client. Another approach used a combination of a transcoding proxy and a content negotiation scheme to optimize the content transmitted to the client based on client advertised capabilities [11]. Our thin client approach differs fundamentally from these fat client approaches by pushing all web browser logic to the server, leveraging existing investments in desktop web browsers to work seamlessly with production systems without any web proxy configuration or web browser modifications.

6. CONCLUSIONS AND FUTURE WORK

We have presented the first experimental study to quantitatively compare the web browsing performance of thin-client systems versus traditional fat clients running native web browsers on wireless PDAs. To make this study possible, we used a variation of slow-motion benchmarking that effectively accounts for end-to-end web browsing latencies in a non-invasive manner. This technique accounts for client processing time during web browsing, which can be significant when using PDAs.

Our measurements demonstrate that thin clients provide better web browsing performance than fat clients across a wide variety of web content, including general consumer content, medical imaging content, and text-based clinical information content widely used in a major academic medical center. Our results show that thin clients can, in some cases, require less bandwidth to achieve superior web browsing performance than fat clients. Our results also show that thin clients can in other cases achieve superior web browsing performance even when they send more data during web browsing. More importantly, our results demonstrate that thin clients provide better web browsing functionality than fat clients running native web browsers on PDAs. While all web page content was viewable using thin clients, several of our experiments demonstrated that PDA web browsers were not able to properly display web page content with any significant JavaScript functionality.

Our results show that thin clients provide faster and more functional web browsing by leveraging existing investments in widely used desktop web browsers and by pushing complex web browsing application logic from less powerful mobile devices to more powerful servers. Our results also show that thin clients can provide faster web browsing than fat clients by clipping the display region on the server before it is sent to the PDA, reducing both client processing time and network bandwidth requirements.

Our study explores two important dimensions of web browsing performance on wireless PDAs, speed and functionality. Another important dimension of performance in the context of PDAs is energy consumption. We have conducted some preliminary studies of energy consumption which indicate that thin clients can extended the battery life of a PDA to last significantly longer than with fat clients. However, the cause of this needs to be investigated further. Given that battery life is a dominant factor in the performance of PDAs, the benefits of thin clients for energy consumption is an important area that merits future work.

7. ACKNOWLEDGMENTS

This research was supported in part by NSF grant CCR-0219943 and an IBM Sur Award. Albert Lai is supported by a National Library of Medicine training grant NO1-LM07079. Klaus Schauser initially suggested that thin clients may outperform fat clients on PDAs, motivating this study. Madhuri Shinde assisted with running the benchmark experiments. Andrew Laine, Yinpeng Jin, and Elizabeth Chen provided the application data for creating some of the benchmarks used in this study.

8. REFERENCES

[1] 80211 planet. http://www.80211-planet.com/.

[2] Virtual Network Computing. `http://www.uk.research.att.com/vnc`.

[3] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan. Characterizing user behavior and network performance in a public wireless lan. In *Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and*

modeling of computer systems, pages 195–205. ACM Press, 2002.

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

[5] E. S. Chen and J. J. Cimino. Automated Discovery of Patient-Specific Clinician Information Needs Using Clinical Information System Log Files. In *Proc. AMIA Symp.*, pages 145–149, Nov. 2003.

[6] J. J. Cimino, S. A. Socratous, and P. D. Clayton. Internet as clinical information system: application development using the world wide web. *J. Am. Med. Inform. Assoc.*, 2(5):273–284, Sept.-Oct. 1995.

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

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

[9] A. Fox, I. Goldberg, S. D. Gribble, and D. C. Lee. Experience with top gun wingman: A proxy-based graphical web browser for the 3com palmpilot. In *Proceedings of Middleware '98, Lake District, England, September 1998*, 1998.

[10] G. Hripcsak, J. J. Cimino, and S. Sengupta. WebCIS: large scale deployment of a Web-based clinical information system. In *Proc. AMIA Symp.*, pages 804–808, 1999.

[11] A. Joshi. On proxy agents, mobility, and web access. *Mobile Networks and Applications*, 5(4):233–241, 2000.

[12] J. Kangasharju, Y. G. Kwon, and A. Ortega. Design and implementation of a soft caching proxy. *Computer Networks and ISDN Systems*, 30(22–23):2113–2121, 1998.

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

[14] T. Keeley. Thin, High Performance Computing over the Internet. In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, page 407, San Francisco, CA, Aug. 2000. IEEE Computer Society.

[15] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, pages 107–118. ACM Press, 2002.

[16] A. Lai and J. Nieh. Limits of Wide-Area Thin-Client Computing. In *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 228–239, Marina del Rey, CA, USA, June 15-19, 2002. ACM Press.

[17] A. Laine, M. Shim, I. Koren, W. Huda, and B. Steinbach. Multiscale Processing Techniques for the Enhancement of Digital Radiographcs. In *Radiological Society of North America (RSNA) 83rd Scientific Assembly and Annual Meeting*, Chicago, IL, USA, Dec. 1997.

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

[19] A. Maheshwari, A. Sharma, K. Ramamritham, and P. Shenoy. Transquid: Transcoding and caching proxy for heterogenous ecommerce environments, 2002.

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

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

[22] Windows 2000 Terminal Services Capacity Planning. Technical White Paper, 2000.

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

[24] J. Nieh, S. J. Yang, and N. Novik. A Comparison of Thin-Client Computing Architectures. Technical Report CUCS-022-00, Department of Computer Science, Columbia University, Nov. 2000.

[25] J. Nieh, S. J. Yang, and N. Novik. Measuring Thin-Client Performance Using Slow-Motion Benchmarking. *ACM Trans. Computer Systems*, 21(1):87–115, Feb. 2003.

[26] J. Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, Indianapolis, Indiana, 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. SCO Technical White Paper, Dec. 1998.

[29] R. W. Scheifler and J. Gettys. The X Window System. *ACM Trans. Gr.*, 5(2):79–106, Apr. 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 (SOSP)*, volume 34, pages 32–47, Kiawah Island Resort, SC, Dec. 1999.

[31] 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 No. 6,104,392, Aug. 2000.

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

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

[34] D. Tang and M. Baker. Analysis of a local-area wireless network. In *Proceedings of the 6th annual International Conference on Mobile Computing and Networking*, pages 1–10. ACM Press, 2000.

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

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

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

[38] S. J. Yang, J. Nieh, S. Krishnappa, A. Mohla, and M. Sajjadpour. Web Browsing Performance of Wireless Thin-Client Computing. In *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, Budapest, Hungary, May 20-24, 2003.

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

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